Sliding Window Temporal Graph Coloring*

George B. Mertzios† Hendrik Molter‡ Viktor Zamaraev§ ¶

Abstract

Graph coloring is one of the most famous computational problems with applications in a wide range of areas such as planning and scheduling, resource allocation, and pattern matching. So far coloring problems are mostly studied on static graphs, which often stand in contrast to practice where data is inherently dynamic. A temporal graph has an edge set that changes over time. We present a natural temporal extension of the classical graph coloring problem. Given a temporal graph and integers k and Δ, we ask for a coloring sequence with at most k colors for each vertex such that in every time window of Δ consecutive time steps, in which an edge is present, this edge is properly colored at least once. We thoroughly investigate the computational complexity of this temporal coloring problem. More specifically, we prove strong computational hardness results, complemented by efficient exact and approximation algorithms.

Keywords: Time-varying Graph, Link Stream, NP-hardness, Parameterized Complexity, Fixed-Parameter Tractability, Channel Assignment.

1 Introduction

A great variety of modern, as well as of traditional networks are dynamic in nature as their link availability changes over time. Just a few indicative examples of such inherently dynamic networks are information and communication networks, social networks, transportation networks, and several physical systems [29, 41]. All these application areas share the common characteristic that the network structure, i.e. the underlying graph topology, is subject to *discrete changes over time*. In this paper, embarking from the foundational work of Kempe et al. [33], we adopt a simple and natural model for time-varying networks, given by a graph with time-labels on its edges, while the vertex set is fixed.

Definition 1.1 (Temporal Graph)**.** *A* temporal graph *is a pair $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$ is an underlying (static) graph and $\lambda : E \to 2^{\mathbb{N}} \setminus \{\emptyset\}$ is a* time-labeling *function which assigns to every edge of G a set of discrete-time labels.*

For every edge $e \in E$ in the underlying graph G of a temporal graph (G, λ), we get that $\lambda(e)$ denotes the set of time slots at which e is *active*. Note that, to avoid trivial cases, we demand in Definition 1.1 that $\lambda(e) \neq \emptyset$ for every edge e. Due to their relevance and applicability in many areas, temporal graphs have been studied from various perspectives and under different names such as *time-varying* [20, 43, 1], *dynamic* [26, 12], *evolving* [8, 19, 15], and *graphs over time* [35]. For a comprehensive overview on the existing models and results on temporal graphs from a (distributed) computing perspective see the surveys [39, 34, 12, 10, 11].

The conceptual shift from static to temporal graphs imposes new challenges in algorithmic computation and complexity. Now the classical computational problems have to be appropriately redefined in the temporal setting in order to properly capture the notion of time. Motivated by the fact that, due to causality, information in temporal graphs can "flow" only along sequences of edges whose time-labels are increasing, most temporal graph parameters and optimization problems

*George B. Mertzios and Viktor Zamaraev are supported by the EPSRC grant EP/P020372/1. Hendrik Molter is supported by the DFG project MATE (NI 369/17). A preliminary conference version of this paper appeared in the proceedings of AAAI 2019 [37]. This version contains full proof details. We are grateful to two anonymous Journal of Computer and System Sciences reviewers whose constructive feedback greatly helped us to improve our presentation.

†Department of Computer Science, Durham University, UK. Email: george.mertzios@durham.ac.uk

‡Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany. Email: h.molter@tu-berlin.de

§Department of Computer Science, University of Liverpool, UK. Email: viktor.zamaraev@liverpool.ac.uk

¶The main part of this paper was prepared while Viktor Zamaraev was affiliated at the Department of Computer Science, Durham University, UK.

that have been studied so far are based on the notion of temporal paths and other "path-related" notions, such as temporal analogues of distance, reachability, separators, diameter, exploration, and centrality [2, 18, 36, 40, 3, 17, 48, 21, 13].

Recently only few attempts have been made to define and study "non-path" temporal graph problems. Motivated by the contact patterns among high-school students, Viard et al. [45], introduced Δ-cliques, an extension of the concept of cliques to temporal graphs (see also [28, 6]). Chen et al. [14] presented an extension of the cluster editing problem to temporal graphs. Furthermore, Akrida et al. [4] introduced the notion of temporal vertex cover, motivated by applications of covering problems in transportation and sensor networks. Temporal extensions of the classical graph coloring problem have also been previously studied by Yu et al. [47] (see also [25]) in the context of channel assignment in mobile wireless networks. In this problem, every edge has to be properly colored in every snapshot of the input temporal graph (G, λ), while the goal is to minimize some linear combination of the total number of colors used and the number of color re-assignments on the vertices [47]. In this temporal coloring approach, the notion of time is only captured by the fact that the number of re-assignments affects the value of the target objective function, while the fundamental solution concept remains the same as in static graph coloring; that is, *every* individual (static) snapshot has to be properly colored. Using this, Yu et al. [47] presented generic methods to adapt known algorithms and heuristics from static graph coloring to deal with their new objective function. Other temporal extensions of the classical vertex and edge coloring problems have been recently studied by Vizing [46]. Vizing considered only temporal graphs of lifetime two, and in his problems every object to color (vertex or edge) has to be colored in exactly one of the snapshots of the input temporal graph in such a way that any two objects that are assigned the same color in the same snapshot are not adjacent in this snapshot. The goal of the problems is to minimize the total number of used colors.

In this paper we introduce and rigorously study a different, yet natural temporal extension of the classical graph coloring problem, called Sliding Window Temporal Coloring (for short, SW-Temporal Coloring). In SW-Temporal Coloring the input is a temporal graph (G, λ) and two natural numbers Δ and k. At every time slot t, every vertex has to be assigned one color, under the following constraint: Every edge e has to be properly colored at least once during *every time window* of Δ consecutive time slots, in which e appears at least once, and this must happen at a time slot t in this window when e is active. Now the question is whether there exists such a temporal coloring over the whole lifetime of the input temporal graph that uses at most k colors. In contrast to the model of Yu et al. [47], the solution concept in SW-Temporal Coloring is fundamentally different to that of static graph coloring as it takes into account the inherent dynamic nature of the network. Indeed, even to verify whether a given solution is feasible, it is not sufficient to just consider every snapshot independently.

Our temporal extension of the static graph coloring problem is motivated by applications in mobile sensor networks and in planning. Consider the following scenario: every mobile agent broadcasts information over a specific communication channel while it listens on all *other* channels. Thus, whenever two mobile agents are sufficiently close, they can exchange information only if they broadcast on different channels. We assume that agents can switch channels at any time. To ensure a high degree of information exchange, it makes sense to find a schedule of assigning broadcasting channels to the agents over time which minimizes the number of necessary channels, while allowing each pair of agents to communicate at least once within every small time window in which they are close to each other.

To further motivate the questions raised in this work, imagine an organization which, in order to ensure compliance with the national laws and the institutional policies, requires its employees to *regularly* undertake special training that is relevant to their role within the organization. Such training requirements can be naturally grouped within training "themes", concerning –for example– the General Data Protection Regulation (GDPR) of the EU for staff dealing with personal data or equality and diversity issues when hiring new employees for Human Resources staff, etc). One reasonable organizational requirement for such a regular staff training is that every employee has to undertake all needed pieces of training at least once within every time-window of a specific length Δ (e.g. $\Delta = 12$ months). All training sessions are offered by experts in predefined "training periods" (e.g. annually every January, May, and September), while each session takes a fixed amount of time to run (e.g. a full day during the corresponding training period). This situation can be naturally modeled as a *temporal* graph problem: (i) each time slot t represents a predefined "training period"; (ii) each vertex v denotes one of the themes that are offered for training by the organization; (iii) the different colors that a vertex v can

take at time slot t represent all different days in which the theme v can be taught during the training period t; (iv) an edge $\{u, v\}$ that is active at the time slot t means that the themes u and v share at least one participant at the corresponding training period. Note that, since the training needs of specific staff members change over time, an edge between two themes u and v may repeatedly appear and disappear over time, and thus the above graph is temporal. If a participant is planned to undertake training on both themes u, v at the same time slot t, then these themes have to run at different days of the time slot t, i.e. u and v have to be assigned different colors at time t. In such a situation, it is natural for the organization to try to schedule all training sessions in such a way that the total *duration* (i.e. number of different colors) of every training period t never exceeds k different days, while simultaneously meeting all regular training requirements.

1.1 Our Contribution

In this paper we introduce the problem Sliding Window Temporal Coloring (for short, SW-Temporal Coloring) and we present a thorough investigation of its computational complexity. All our notation and the formal definition of the temporal problems that we study are presented in Section 2. First we investigate in Section 3 an interesting special case of SW-Temporal Coloring, called Temporal Coloring, where the length Δ of the sliding time window is equal to the whole lifetime T of the input temporal graph. We start by proving in Theorem 3.1 that Temporal Coloring is NP-hard even for $k = 2$, and even when every time slot consists of one clique and isolated vertices. This is in wide contrast to the static coloring problem, where it can be decided in linear time whether a given (static) graph G is 2-colorable, i.e. whether G is bipartite. On the positive side, we show in Theorem 3.4 that, given any input temporal graph (G, λ) for Temporal Coloring with n vertices and lifetime T, we can compute an equivalent instance (G', λ') on the same vertices but with lifetime $T' \leq m$, where m is the number of edges in the underlying graph G. Moreover we show that the new instance can be computed in polynomial time. Formally, Theorem 3.4 shows that Temporal Coloring admits a polynomial kernel when parameterized by the number n of vertices of the input temporal graph. That is, we can efficiently preprocess any instance of Temporal Coloring to obtain an equivalent instance whose size only depends polynomially on the size of the underlying graph G and not on the lifetime T of (G, λ).

In Section 4 and in the remainder of the paper we deal with the general version of SW-Temporal Coloring, where the value of Δ is arbitrary. On the one hand, we show that the problem is hard even on very restricted special classes of input temporal graphs. On the other hand, assuming the Exponential Time Hypothesis (ETH), we give an asymptotically optimal exponential-time algorithm for SW-Temporal Coloring whenever Δ is constant. Moreover we show how to extend it to get an algorithm which runs in linear time if the number n of vertices is constant. Note here that the *size* of the input temporal graph also depends on its lifetime T whose value can still be arbitrarily large, independently of n. Furthermore note that this assumption about n being a constant can be also reasonable in practical situations; for example, in our motivation above about planning the training of staff in an organization, the value of n equals the number of different "training themes" to be run, which can be expected to be rather small.

Finally we consider in Section 4 an optimization variant of SW-Temporal Coloring where the number of colors is to be minimized. We give an approximation algorithm with an additive error of 1 which runs in linear time on instances where the underlying graph G of the input temporal graph (G, λ) has a constant-size vertex cover. From a classification standpoint this is also optimal since the problem remains NP-hard to solve optimally on temporal graphs where the underlying graph has a constant-size vertex cover.

2 Preliminaries and Notation

Given a (static) graph G, we denote by $V(G)$ and $E(G)$ the sets of its vertices and edges, respectively. An edge between two vertices u and v of G is denoted by $\{u, v\}$, and in this case u and v are said to be *adjacent* in G. A *complete graph* is a graph where every pair of vertices is adjacent. For simplicity of presentation we may refer to a complete graph also as a *clique*. The complete graph on n vertices is denoted by K_n. For every $i, j \in \mathbb{N}$, where $i \leq j$, we let $[i, j] = \{i, i+1, \ldots, j\}$ and $[j] = [1, j]$. Throughout the paper we consider temporal graphs with *finite lifetime* T, that is, there is a maximum label assigned

by λ to an edge of G, called the *lifetime* of (G, λ); it is denoted by $T(G, \lambda)$, or simply by T when no confusion arises. Formally, $T(G, \lambda) = \max\{t \in \lambda(e) : e \in E\}$. We refer to each integer $t \in [T]$ as a *time slot* of (G, λ). The *instance* (or *snapshot*) of (G, λ) *at time* t is the static graph $G_t = (V, E_t)$, where $E_t = \{e \in E : t \in \lambda(e)\}$. If $E_t = \emptyset$, we call $G_t = (V, E_t)$ a *trivial snapshot*. For every subset $S \subseteq [T]$ of time slots, we denote by $E[S] = \bigcup_{i \in S} E_i$ the union of all edges appearing in at least one of the time slots in the set S. Furthermore, we denote by $(G, \lambda)|_S$ the restriction of (G, λ) to the time slots in S. In particular, for the case where $S = [i, j]$ for some $i, j \in [T]$, where $i \leq j$, we have that $(G, \lambda)|_{[i,j]}$ is the sequence of the instances $G_i, G_{i+1}, \ldots, G_j$. We assume in the remainder of the paper that every edge of G appears in at least one time slot until T, namely $\bigcup_{t=1}^{T} E_t = E$.

In the remainder of the paper we denote by $n = |V|$ and $m = |E|$ the number of vertices and edges of the underlying graph G, respectively, unless otherwise stated. Furthermore, unless otherwise stated, we assume that the labeling λ is arbitrary, i.e. (G, λ) is given with an explicit list of labels for every edge. That is, the *size* of the input temporal graph (G, λ) is $O\left(|V| + \sum_{t=1}^{T} |E_t|\right) = O(n + mT)$. In other cases, where λ is more restricted, e.g. if λ is periodic or follows another specific temporal pattern, there may exist more succinct representations of the input temporal graph.

For every $v \in V$ and every time slot t, we denote the *appearance of vertex* v *at time* t by the pair (v, t). That is, every vertex v has T different appearances (one for each time slot) during the lifetime of (G, λ). For every time slot $t \in [T]$ we denote by $V_t = \{(v, t) : v \in V\}$ the set of all vertex appearances of (G, λ) at the time slot t. Note that the set of all vertex appearances in (G, λ) is the set $V \times [T] = \cup_{1 \leq t \leq T} V_t$.

2.1 Temporal Coloring

A *temporal coloring* of a temporal graph $\mathcal{G} = (G, \lambda)$ is a function $\Upsilon : V \times [T] \to \mathbb{N}$, which assigns to every vertex appearance (v, t) with $t \in [T]$ in $\mathcal{G}$ one color $\Upsilon(v, t) \in \mathbb{N}$. The *size* of Υ is the total number $|\Upsilon| := |\bigcup_{v \in V, t \in [T]} \{\Upsilon(v, t)\}|$ of colors used by Υ. For every time slot $t \in [T]$ we denote by Υ_t the restriction of Υ to the vertex appearances at time slot t, that is, $\Upsilon_t : V \to \mathbb{N}$, such that $\Upsilon_t(v) = \Upsilon(v, t)$, for every $v \in V$. We refer to Υ_t as the *time slot coloring* for the time slot t. Furthermore, to ease the presentation, we will refer to the temporal coloring Υ as the ordered sequence $(\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_T)$ of all its time slot colorings. Let $e \in E(G)$ be an edge of the underlying graph G. We say that an edge $e = \{u, v\}$ of the underlying graph G is *properly temporally colored* at time slot t if $\Upsilon_t(u) \neq \Upsilon_t(v)$ and $e \in E_t$, that is, the edge e is present at time slot t. We are now ready to introduce the definition of a *proper temporal coloring*.

Definition 2.1 (Proper Temporal Coloring)**.** *Let $\mathcal{G} = (G, \lambda)$ be a temporal graph. A* proper temporal coloring *of $\mathcal{G}$ is a temporal coloring $\Upsilon = (\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_T)$ such that every edge $e \in E(G)$ is properly temporally colored in at least one time slot $t \in [T]$.*

Using this definition, we can formally define the decision problem Temporal Coloring.

Temporal Coloring	
Input:	A temporal graph $\mathcal{G} = (G, \lambda)$ and an integer $k \in \mathbb{N}$.
Question:	Is there a proper temporal coloring Υ of $\mathcal{G}$ using $\|\Upsilon\| \leq k$ colors?

Note that Temporal Coloring is a natural extension of the classic NP-complete Coloring problem [32, 24, 23] on static graphs to temporal graphs. In particular, Coloring is the special case of Temporal Coloring where the lifetime of the input temporal graph is $T = 1$. Moreover, it is easy to see that it can be verified in polynomial time whether a given temporal coloring Υ is proper. Hence, we have that Temporal Coloring is NP-complete for each fixed $k \geq 3$ and $T \geq 1$ (since Coloring is NP-complete for all $k \geq 3$ [24, 23] and to get $T > 1$ we can add trivial snapshots to the temporal graph).

We remark that Temporal Coloring can be treated as a *multi-layer graph* problem: It is easy to check that, given a temporal graph $\mathcal{G} = (G, \lambda)$ and an integer $k \in \mathbb{N}$, it holds that for every permutation $\pi : [T] \to [T]$ we have that $(\mathcal{G}, k)$ is a YES-instance if and only if $(\mathcal{G}' = (V, (E_{\pi(i)})_{i \in [T]}), k)$ is a YES-instance.

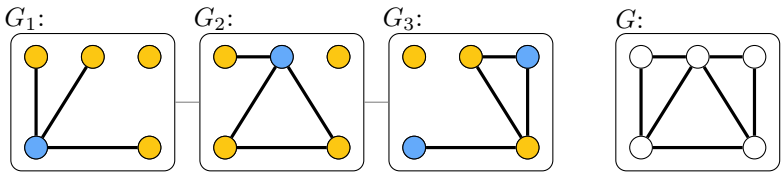


Figure 1: Example temporal graph with lifetime three and a proper sliding Δ-window temporal 2-coloring for $\Delta = 2$. Notice that for example one edge of G_2 is colored monochromatically, but this edge is also active at time slots one and three and is colored properly in the corresponding snapshots.

2.2 Sliding Δ-Window Temporal Coloring

In the definition of a proper temporal coloring given in Definition 2.1, we require that every edge is properly temporally colored at least once during the whole lifetime T of the temporal graph $\mathcal{G}$. However, in many real-world applications, where T is expected to be arbitrarily large, we may need to require that every edge is properly temporally colored more often, and in particular, at least once during every Δ-time window in which it appears, for some given Δ, regardless of how large the lifetime T is. We formalize this in the definition of a *proper sliding Δ-window temporal coloring*, where we denote by $W_t^\Delta = [t, t+\Delta-1]$ the Δ-window starting at time slot t.

Definition 2.2 (Proper Sliding Δ-Window Temporal Coloring)**.** *Let $\mathcal{G} = (G, \lambda)$ be a temporal graph and let $\Delta \leq T$. A* proper sliding Δ-window temporal coloring *of $\mathcal{G}$ is a temporal coloring $\Upsilon = (\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_T)$ such that, for every Δ-window W_t^Δ with $t \in [T-\Delta+1]$ and for every edge $e \in E[W_t^\Delta]$ we have that e is properly temporally colored in at least one time slot $t' \in W_t^\Delta$.*

An example of a proper sliding Δ-window temporal coloring is given in Figure 1. Using this definition, we can formally define the decision problem SW-Temporal Coloring.

SW-Temporal Coloring
Input: A temporal graph $\mathcal{G} = (G, \lambda)$ and two integers $k \in \mathbb{N}$ and $\Delta \leq T$.
Question: Is there a proper sliding Δ-window temporal coloring Υ of $\mathcal{G}$ using $|\Upsilon| \leq k$ colors?

Note that the problem Temporal Coloring defined above in this section is the special case of SW-Temporal Coloring where $\Delta = T$, that is, where there is only one Δ-window in the whole temporal graph. Moreover, it is easy to see that is can be verified in polynomial time whether a given temporal coloring Υ is a proper sliding Δ-window temporal coloring. Hence, we have that SW-Temporal Coloring is NP-complete for each fixed $k \geq 3$, $\Delta \geq 1$, and $T \geq \Delta$. Moreover note that, in contrast to Temporal Coloring (i.e. the problem version without a sliding window), SW-Temporal Coloring cannot be treated any more as a multi-layer graph problem. In fact, as it can be easily checked, the answer to SW-Temporal Coloring may change if we modify the ordering of the snapshots of the input temporal graph.

2.3 Basic Observations

We start with the observation that computational hardness of SW-Temporal Coloring for some fixed value of Δ implies hardness for all larger values of Δ. This allows us to construct hardness reductions for small fixed values of Δ and still obtain general hardness results.

Observation 2.3. *Let Δ be a fixed constant.* SW-Temporal Coloring *on instances $(\mathcal{G}, k, \Delta+1)$ is computationally at least as hard as* SW-Temporal Coloring *on instances $(\mathcal{G}, k, \Delta)$.*

Proof. To see the correctness of Observation 2.3, we show that we can easily reduce from SW-Temporal Coloring with input Δ to SW-Temporal Coloring with input $(\Delta+1)$ by inserting a trivial snapshot after every Δ consecutive snapshots. Let $(\mathcal{G}, k, \Delta)$ denote the original instance and $(\mathcal{G}', k, \Delta+1)$ the constructed instance. For a visualization see Figure 2.

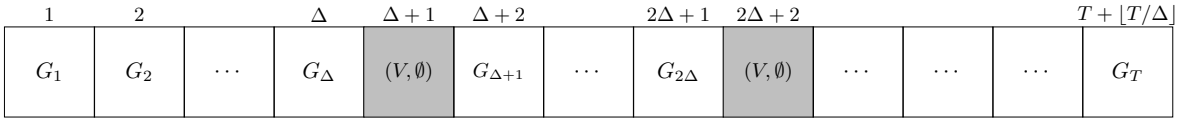


Figure 2: Inserting trivial snapshots to reduce SW-Temporal Coloring on instances $(\mathcal{G}, \Delta, k)$ to SW-Temporal Coloring on instances $(\mathcal{G}, \Delta + 1, k)$.

($\Rightarrow$): If $\mathcal{G}$ admits a proper sliding Δ-window temporal coloring, then we can easily modify this coloring for $\mathcal{G}'$. The inserted trivial snapshots can be colored arbitrarily and all other snapshots are colored in the same way the corresponding snapshots from $\mathcal{G}$ are colored. This yields a proper sliding $(\Delta + 1)$-window temporal coloring for $\mathcal{G}'$.

($\Leftarrow$): If $\mathcal{G}'$ admits a proper sliding $(\Delta + 1)$-window temporal coloring, then we can easily modify this coloring for $\mathcal{G}$. We ignore how the inserted trivial snapshots are colored and color all snapshots of $\mathcal{G}$ in the same way the corresponding snapshots from $\mathcal{G}'$ are colored. This yields a proper sliding Δ-window temporal coloring for $\mathcal{G}$. □

2.4 Parameterized complexity

We use standard notation and terminology from parameterized complexity [16]. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. We call the second component the *parameter* of the problem. A parameterized problem is *fixed-parameter tractable* (in the complexity class FPT) if there is an algorithm that solves each instance (I, r) in $f(r) \cdot |I|^{O(1)}$ time, for some computable function f. A parameterized problem L admits a *polynomial kernel* if there is a polynomial-time algorithm that transforms each instance (I, r) into an instance (I', r') such that $(I, r) \in L$ if and only if $(I', r') \in L$ and $|(I', r')| \leq r^{O(1)}$.

In the following, we give definitions of the graph parameters we consider in this paper. The *degeneracy* of a graph G is the smallest integer $d \in \mathbb{N}$ such that each subgraph G' of G contains a vertex v with degree at most d. The *domination number* of a graph $G = (V, E)$ is the cardinality of a minimum vertex subset $V' \subseteq V$ such that every vertex $v \in V$ is either contained in V' or has a neighbor that is contained in V'. The *vertex cover number* of a graph $G = (V, E)$ is the cardinality of a minimum vertex subset $V' \subseteq V$ such that every edge $e \in E$ has at least one endpoint in V'. Note that most of the mentioned parameters also have equivalent alternative definitions.

3 Temporal Coloring

In this section we investigate the parameterized computational complexity of Temporal Coloring. We give two hardness results that in particular show that Temporal Coloring is already NP-complete for two colors, even if the input temporal graph is very restricted. This stands in stark contrast to the static case, where checking whether a graph is 2-colorable can be done in linear time.

On the algorithmic side, we show that Temporal Coloring admits a polynomial kernel when parameterized by the number of vertices of the input temporal graph.

3.1 Refined NP-Hardness Results

We start by showing that Temporal Coloring is NP-complete even if each snapshot consists of a clique together with some isolated vertices. From a motivation standpoint, this excludes an interesting special case of the mobile agent scenario, where at each time slot exactly one group of agents meet such that they can all pairwise communicate.

Theorem 3.1. Temporal Coloring *is NP-complete for each fixed $k \geq 2$ even if each snapshot consists of one clique together with isolated vertices.*

We show this result for $k = 2$ and later explain why our proof is easily adaptable for larger values of k. First we show the following lemma, which we will make use of in the proof of Theorem 3.1.

Lemma 3.2. *A graph G has two bipartite subgraphs that cover all edges of G if and only if G is 4-colorable.*

Proof. Assume that a given graph $G = (V, E)$ has two bipartite subgraphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ that cover all edges E of G, that is, $E = E_1 \cup E_2$. Let $\Upsilon_i : V \to \{1, 2\}$ be the coloring of G_i for $i \in \{1, 2\}$. Then $\Upsilon(v) := \pi(\Upsilon_1(v), \Upsilon_2(v))$ is a 4-coloring for G, where π is an arbitrary pairing function[1]: First note that $|\bigcup_{v \in V} \{\Upsilon(v)\}| \leq 4$ since $\Upsilon(v) \in \{\pi(1,1), \pi(1,2), \pi(2,1), \pi(2,2)\}$ for all $v \in V$. Now let $\{v, w\} \in E$. By assumption we have that $\{v, w\} \in E_1 \cup E_2$. Assume that $\{v, w\} \in E_1$ (the other case is analogous), then we have that $\Upsilon_1(v) \neq \Upsilon_1(w)$. It follows that $\Upsilon(v) \neq \Upsilon(w)$.

It remains to show that if a given graph $G = (V, E)$ is 4-colorable, then it has two bipartite subgraphs that cover all edges of G. Let $\Upsilon : V \to \{1, 2, 3, 4\}$ be a 4-coloring for G. Let $E_1 := \{\{v, w\} \in E \mid \Upsilon(v) \in \{1, 2\} \wedge \Upsilon(w) \in \{3, 4\}\}$ and $E_2 := \{\{v, w\} \in E \mid \Upsilon(v) \in \{1, 3\} \wedge \Upsilon(w) \in \{2, 4\}\}$. It is easy to check that $G_1 = (V, E_1)$ is bipartite: One part is formed by vertices colored in 1 or 2 and the second part by vertices colored in 3 or 4. By definition E_1 does not contain edges between vertices from the same part. The argument for $G_2 = (V, E_2)$ is analogous. They are both subgraphs since $E_i \subseteq E$ for $i \in \{1, 2\}$. It remains to show that $E = E_1 \cup E_2$: Let $\{v, w\} \in E$, then if $\{\Upsilon(v), \Upsilon(w)\} \in \{\{1, 3\}, \{2, 3\}, \{1, 4\}, \{2, 4\}\}$, then $\{v, w\} \in E_1$, otherwise (if $\{\Upsilon(v), \Upsilon(w)\} \in \{\{1, 2\}, \{3, 4\}\}$) we have that $\{v, w\} \in E_2$. □

Now we prove Theorem 3.1 for the case that $k = 2$.

Proof of Theorem 3.1 for $k = 2$. We give a reduction from the NP-complete 4-Coloring problem [24, 23] where, given a graph H, we are asked whether a proper 4-coloring for H exists. Let $H = (U, F)$ be an instance of 4-Coloring. We construct a temporal graph $\mathcal{G} = (G, \lambda)$ with $V = U$, $E_1 = E_2 = E(K_{|U|})$, $T = \binom{|U|}{2} - |F| + 2$, and for every non-edge of H there is exactly one time slot i with $3 \leq i \leq T$ where only this edge is present. Note that every snapshot is either complete or only contains a single edge. Hence, every snapshot consists of a clique together with some isolated vertices.

Correctness. We now prove the correctness of our reduction, namely, show that the constructed temporal graph can be properly temporally colored with two colors if and only if the input graph is 4-colorable.

($\Rightarrow$): If H is 4-colorable, then we can use the 4-coloring of H to 2-color G_1 and G_2 using Lemma 3.2 and for every edge that is not present in G, color it properly in the snapshot where it is present.

($\Leftarrow$): If $\mathcal{G}$ is properly colorable with $k = 2$ colors, then all edges present in H have to be properly colored either in G_1 or G_2, that is, the edges of H can be covered by two bipartite graphs, and hence, by Lemma 3.2, we can properly 4-color H. □

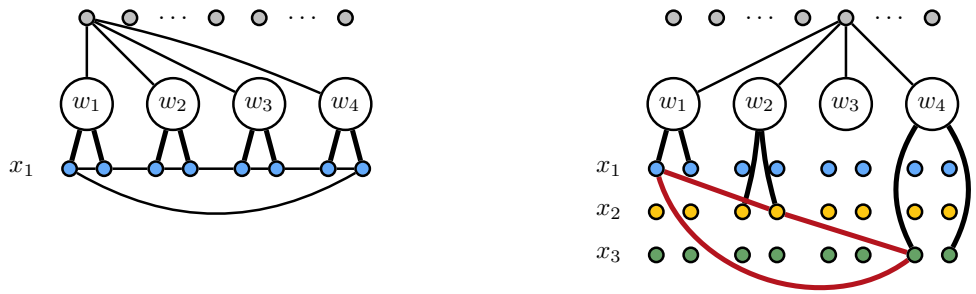
To adapt this proof for larger values of k, it is necessary to generalize Lemma 3.2 to the statement "A graph G has two k-colorable subgraphs that cover all edges of G if and only if G is k^2-colorable". It is easy to check that this can be done in an analogous way for each fixed k. Using this more general lemma, one can easily adapt the reduction in the proof of Theorem 3.1. We remark that, from a parameterized point of view, this result implies that parameterizing Temporal Coloring by structural graph parameters of the snapshots that are constant on a graph consisting of a clique with some isolated vertices cannot yield fixed-parameter tractability unless P = NP, even if combined with k.

Now we show with a more refined reduction that Temporal Coloring remains hard even if each snapshot has very few edges and the underlying graph has small degeneracy.

Theorem 3.3. Temporal Coloring *is NP-complete for all $k \geq 2$ even if the number of edges in each snapshot is in $O(k^2)$, the degeneracy of the underlying graph is in $O(k)$ and every snapshot as well as the underlying graph has domination number four.*

Proof. We present a reduction from Exact $(3, 4)$-SAT [44] to Temporal Coloring with $k = 2$. The reduction can be easily modified to a larger number of colors, we explain how to do this at the end of the proof. In Exact $(3, 4)$-SAT we are asked to decide whether a given Boolean formula ϕ is satisfiable and ϕ is in conjunctive normal form where every clause has exactly three distinct literals and every variable appears in exactly four clauses. Given a formula ϕ with n variables and m clauses, we construct a temporal graph $\mathcal{G} = (G, \lambda)$ consisting of $T = (n+2m)$ snapshots, that is, one snapshot for each variable gadget and two snapshots for each clause gadget. An illustration of the construction is given in Figure 3. We start by adding four vertices w_1, w_2, w_3, and w_4 which will help to encode the first, second, third,

[1] A function $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a *pairing function* if it is a bijection.



(a) Variable gadget.

(b) First snapshot of clause gadget.

Figure 3: Illustration of the reduction from Exact $(3,4)$-SAT to Temporal Coloring of the proof of Theorem 3.3. Figure 3a depicts the variable gadget for x_1. Figure 3b depicts the first snapshot of the clause gadget for clause $(x_1 \vee \neg x_2 \vee x_3)$, where we have the first appearance of x_1 (blue), the second appearance of x_2 (yellow), and the fourth appearance of x_3 (green). The second snapshot of the clause gadget contains only the red triangle and is not depicted. In both figures vertices corresponding to the remaining variables are not depicted. Thick edges are present in exactly two snapshots and thin edges are present in exactly one snapshot.

and fourth appearance of a variable.

Variable gadget. For each variable x_i with $i \in [n]$ of ϕ we create nine vertices $v_{x_i}^{(1)}, v_{x_i}^{(2)}, \ldots, v_{x_i}^{(8)}$ (which we also refer to as "the vertices corresponding to x_i"), and u_{x_i}, and one new snapshot. In this new snapshot, we connect $v_{x_i}^{(j)}$ with $v_{x_i}^{((j \bmod 8)+1)}$ for all $j \in [8]$ and we connect $v_{x_i}^{(2h-1)}$ and $v_{x_i}^{(2h)}$ with w_h for all $h \in [4]$. Furthermore, we connect u_{x_i} with w_1, w_2, w_3, and w_4. It is easy to check that every snapshot corresponding to a variable contains twenty edges.

Clause gadget. For each clause c_i with $1 \leq i \leq m$ of ϕ we add two new snapshots and one new vertex u_{c_i}. In the first new snapshot we connect it with w_1, w_2, w_3, and w_4. Let x_j be a variable that appears in clause c_i and let this be the hth appearance of x_j in ϕ. Then we connect w_h with $v_{x_j}^{(2h-1)}$ and $v_{x_j}^{(2h)}$ in the first new snapshot. Lastly, denote x_{j_1}, x_{j_2}, and x_{j_3} the three variables in c_i appearing for the h_1th, h_2th, and h_3th time, respectively, and let $y_s = 1$ if x_{j_s} appears non-negated in c_i and $y_s = 0$, otherwise. We pairwise connect $v_{x_{j_1}}^{(2h_1-y_1)}$, $v_{x_{j_2}}^{(2h_2-y_2)}$, and $v_{x_{j_3}}^{(2h_3-y_3)}$ in both the first and the second new snapshot, we refer to these three vertices as "the triangle corresponding to clause c_i". It is easy to check that every snapshot corresponding to a clause contains at most thirteen edges.

Before we show correctness, let us check that the underlying graph of $\mathcal{G}$ has constant degeneracy. We can show this by using the following degeneracy ordering: First we order all vertices u_{x_i} and u_{c_i} arbitrarily and put them at the beginning of the degeneracy ordering. Then we order the vertices $v_{x_i}^{(j)}$ arbitrarily and put them next in the ordering. Lastly, we add vertices w_1, w_2, w_3, and w_4 to the ordering. Note that all vertices u_{x_i} and u_{c_i} for some variable x_i or clause c_i, respectively, have degree four since they are only connected to w_1, w_2, w_3, and w_4. The vertices $v_{x_i}^{(j)}$ have degree five: In the snapshot of the variable gadget for x_i they are connected to two other vertices $v_{x_i}^{(j')}$ and $v_{x_i}^{(j'')}$ and to one of the vertices w_1, w_2, w_3, and w_4. Then, depending on the value of j, there is at most one "clause triangle" that contains $v_{x_i}^{(j)}$. Once all these vertices are removed from the graph, the vertices w_1, w_2, w_3, and w_4 are isolated. It follows that the degeneracy of G is at most five.

Furthermore, it is straightforward to check that the vertices w_1, w_2, w_3, and w_4 form a dominating set in every snapshot as well as the underlying graph.

Correctness. It is easy to check that the reduction can be computed in polynomial time. It remains to show that $\mathcal{G}$ admits a proper temporal 2-coloring if and only if ϕ is satisfiable.

($\Rightarrow$): Assume that we are given a satisfying assignment for ϕ. Then we construct a proper temporal 2-coloring for $\mathcal{G}$ as follows. Let the two colors be red and blue. Whenever we do not specify the color of vertices in a certain snapshot, those vertices can be colored arbitrarily in that snapshot. In each snapshot, we color all vertices u_{x_i} and u_{c_j} with $i \in [n]$ and $j \in [m]$ red and vertices w_1, w_2, w_3, and w_4

blue.

Now consider the snapshots corresponding to variable gadgets. If variable x_i is set to true in the satisfying assignment for ϕ, then we color (in the snapshot corresponding to the variable gadget for x_i) vertices $v_{x_i}^{(2h-1)}$ red and vertices $v_{x_i}^{(2h)}$ blue for $h \in [4]$. Otherwise we color the vertices exactly in the opposite way. This leaves exactly four edges monochromatic in each snapshot corresponding to a variable gadget. These will be colored properly in the four clause gadgets corresponding to the four clauses where the corresponding variable appears.

Next, consider the snapshots corresponding to clause gadgets, in particular the first snapshot corresponding to clause c_i. Let x_1, x_2, and x_3 be the three variables appearing in c_i and, without loss of generality, let x_1 be contained in a literal that satisfies the clause and let that be the hth appearance of x_1. If x_1 appears non-negated, then we color $v_{x_1}^{(2h-1)}$ blue and all other vertices corresponding to variables x_1, x_2, and x_3 red. Otherwise, we color $v_{x_1}^{(2h)}$ blue and all other vertices corresponding to variables x_1, x_2, and x_3 red. Since the literal containing x_1 satisfies clause c_i, we have that the edge between w_h and $v_{x_1}^{(2h-1)}$ or $v_{x_1}^{(2h)}$, respectively, is colored properly in the snapshot corresponding to the variable gadget of x_1. Hence all edges between w_1, w_2, w_3, w_4 and vertices corresponding to variables x_1, x_2, and x_3 are colored properly. Out of the edges that form the triangle corresponding to c_i in the snapshot corresponding to clause c_i, exactly one is colored monochromatically. We color the vertices of the triangle in the *second* snapshot corresponding to the variable clause of c_i such that exactly that edge is colored properly. It is easy to verify that this describes a proper temporal 2-coloring for $\mathcal{G}$.

($\Leftarrow$): Assume that we are given a proper temporal 2-coloring for $\mathcal{G}$. Then we construct a satisfying assignment for ϕ in the following way: We start with the observation that in any proper temporal coloring, vertices w_1, w_2, w_3, and w_4 have the same color in each snapshot that corresponds to a variable gadget and in each first snapshot corresponding to a clause gadget. Further, in each snapshot corresponding to a variable gadget there is a cycle of size eight containing all vertices corresponding to the variable of that gadget. Let that variable be x_i. Since all edges involved in this cycle only exists in this one snapshot, there are exactly two ways to color this cycle. One of them leaves the edges between $v_{x_i}^{(2h-1)}$ and w_h monochromatic for $h \in [4]$. The other way to color the cycle is the inverse coloring and leaves the edges between $v_{x_i}^{(2h)}$ and w_h monochromatic for $h \in [4]$. In the first case, we set x_i to false, and in the second case we set x_i to true. We claim that this yields a satisfying assignment for ϕ.

Assume for contradiction that it is not. Then there is a clause that is not satisfied. Let that clause be c_i. Recall that in a proper coloring, also vertices w_1, w_2, w_3, and w_4 have the same colors in each first snapshot that corresponds to a clause gadget. Consider the triangle corresponding to clause c_i in the first snapshot of the clause gadget of c_i. We have that in a proper temporal coloring, this triangle cannot be monochromatic, since, otherwise, one of the three edges is not properly colored in any of the snapshots of the temporal graph. Note that the triangle edges only exist in the two snapshots corresponding to the clause gadget of c_i and in the second snapshot, not all three edges can be colored properly. Hence, in the first snapshot of the clause gadget, at least one of the vertices of the triangle corresponding to c_i has a different color than vertices w_1, w_2, w_3, and w_4. However, this means that the variable corresponding to this particular vertex (i.e., the vertex with the different color) is set to a truth value that satisfies this clause—a contradiction.

Modification for a Larger Number of Colors. To modify this reduction for more colors we introduce new vertices and edges to all snapshots to "block" all colors except two from being used. Formally, we do the following. Let $k > 2$. For each snapshot $i \in [T]$, we add $k-2$ fresh vertices $c_1^{(i)}, \ldots, c_{k-2}^{(i)}$, connect them to form a clique in snapshot i, and connect them to all non-isolated vertices in snapshot i. In all snapshots different from i the vertices $c_1^{(i)}, \ldots, c_{k-2}^{(i)}$ are isolated. All new edges exist in exactly one snapshot and hence have to be colored properly in this snapshot. It follows that the vertices $c_1^{(i)}, \ldots, c_{k-2}^{(i)}$ have to be colored with $k-2$ distinct colors and these colors cannot be used to color any other non-isolated vertex in snapshot i.

The modification introduces $T \cdot (k-2)$ new vertices to the temporal graph and it is easy to check that it introduces $O(k^2)$ new edges to each snapshot. The degeneracy of the underlying graph is in $O(k)$ since we can put all new vertices to the beginning of the degeneracy ordering described earlier in this proof, and it is easy to check that all new vertices have degree in $O(k)$ in the underlying graph. Vertices w_1, w_2, w_3, and w_4 still form a dominating set in every snapshot as well as the underlying graph since they are connected to all new vertices. □

We remark that Theorem 3.3 has some interesting implications from a parameterized point of view. Parameterizing Temporal Coloring by structural graph parameters of the snapshots which are constant on graphs that contain only constantly many edges cannot yield fixed-parameter tractability unless P = NP, even if combined with k. Note that almost all popular structural parameters fall into this category, such as for example "vertex cover number", "feedback edge set", "maximum degree" and all structurally smaller parameters (including for example "treewidth" and "degeneracy").

3.2 Polynomial Kernel for Temporal Coloring

We prove that, given a temporal graph $\mathcal{G} = (G, \lambda)$ for Temporal Coloring with $n = |V(G)|$ vertices, $m = |E(G)|$ underlying edges, and lifetime T, we can efficiently compute an equivalent instance $\mathcal{G}' = (G', \lambda')$ with $T' \leq m$ time slots. The main idea is that if we have sufficiently many time slots, then every edge can be colored in its own time slot and any excess time slots can be removed (as they could be colored arbitrarily). Formally, Theorem 3.4 shows that Temporal Coloring admits a polynomial kernel when parameterized by the number n of vertices.

Theorem 3.4. *Let $\mathcal{G} = (G, \lambda)$ be a temporal graph of lifetime T. Then there exists a temporal graph $\mathcal{G}' = (G', \lambda') = (G, \lambda)|_S$ for some $S \subseteq [T]$ with $|S| \leq m = |E(G)|$ such that for any $k \geq 2$ we have that $\mathcal{G}$ admits a proper temporal k-coloring if and only if $\mathcal{G}'$ admits a proper temporal k-coloring. Furthermore, $\mathcal{G}'$ can be constructed in $O(mT\sqrt{m+T})$ time.*

Proof. Let $\mathcal{G} = (G, \lambda)$ be a temporal graph with lifetime T. If $T \leq m$, then we let $\mathcal{G}' = (G', \lambda') = (G, \lambda)$ and we are done. From now on we assume that $T > m$. We define $B_{(G,\lambda)}$ to be the bipartite graph with two parts $E = E(G)$ and $[T]$, and the edge set $L = \{(e, t) \mid e \in E, t \in [T], t \in \lambda(e)\}$, that is, $e \in E$ is adjacent to $t \in [T]$ if and only if e appears in time slot t in (G, λ). Let $M = \{(e_1, t_1), (e_2, t_2), \ldots, (e_s, t_s)\}$ be a maximum matching in $B = B_{(G,\lambda)}$. We claim that $(G', \lambda') = (G, \lambda)|_{\{t_1, t_2, \ldots, t_s\}}$ admits a proper temporal k-coloring if and only if (G, λ) admits a proper temporal k-coloring.

Given a set $M' \subseteq M$ let $E_{M'} = \{e \mid (e, t) \in M'\}$ and $S_{M'} = \{t \mid (e, t) \in M'\}$. Let $M_1 \subseteq M$ be the set of edges such that every vertex in E_{M_1} is reachable from a vertex in $\overline{E_M} = E \setminus E_M$ by an M-alternating path, i.e. a path whose edges belong alternately to M and not to M. Let $M_2 = M \setminus M_1$.

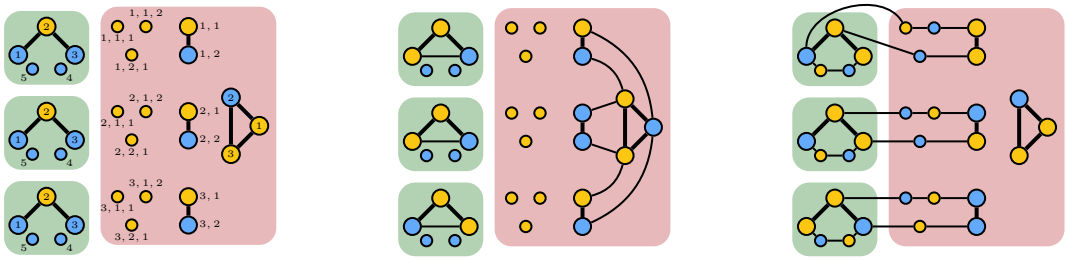
We claim that no vertex in $E_{M_1} \cup \overline{E_M}$ has a neighbour outside S_{M_1}. First, a vertex $e \in E_{M_1} \cup \overline{E_M}$ does not have a neighbour in $\overline{S_M} = [T] \setminus S_M$, as otherwise there would exist an M-augmenting path in B, contradicting the maximality of M. Also, a vertex $e \in E_{M_1} \cup \overline{E_M}$ is not adjacent to a vertex $t_j \in S_{M_2}$, as otherwise the corresponding matching neighbour e_j of t_j would be reachable by an M-augmenting path from a vertex in $\overline{E_M}$, which would contradict the fact that (e_j, t_j) belongs to M_2.

The above claim means that those edges of G that are in $E_{M_1} \cup \overline{E_M}$ appear, and therefore can be properly colored, only in time slots in S_{M_1}. Furthermore, all the edges in E_{M_2} can be properly colored with 2 colors in slots in S_{M_2}: every edge $e \in E_{M_2}$ can be properly colored in the separate time slot t, where $(e, t) \in M_2$. This implies that that (G', λ') admits a proper temporal k-coloring if and only if (G, λ) admits a proper temporal k-coloring, as required.

We can construct the graph $B = B_{(G,\lambda)}$ in $O(mT)$ time, and find a maximum matching M in B in $O(mT\sqrt{m+T})$ time [38]. □

4 SW-Temporal Coloring

In this section we investigate the parameterized computational complexity of SW-Temporal Coloring. We first give a refined NP-hardness reduction together with an ETH lower bound. We give an exponential time algorithm that matches the lower bound for constant Δ and show how to extend this algorithm to obtain fixed-parameter tractability for SW-Temporal Coloring when parameterized by the number of vertices of the input temporal graph. In contrast to Temporal Coloring we can show that SW-Temporal Coloring does not admit a polynomial kernel when parameterized by the number of vertices of the input temporal graph unless NP $\subseteq$ coNP/poly. We proceed by showing that SW-Temporal Coloring is NP-complete even if $k = 2$ and the underlying graph of the input temporal graph has a vertex cover number that only depends on k. Lastly, we show how to adapt our algorithm for SW-Temporal Coloring for a canonical optimization variant of the problem, where we want to minimize the number of colors. We achieve an FPT-approximation algorithm that uses at most one



(a) Snapshot one. (b) Snapshot two. (c) Snapshot three.

Figure 4: Illustration of the reduction from Exact $(3,4)$-SAT to SW-Temporal Coloring of the proof of Theorem 4.1. Vertices and edges in the red shaded areas (right) correspond to a clause gadget for clause $(\neg x_1 \vee x_2 \vee x_3)$. Vertices and edges in the green shaded areas (left) correspond to the variable gadgets for x_1, x_2, and x_3. Thick edges appear in every snapshot while thin edges only appear in one snapshot. The vertices are colored according to a coloring that would be constructed for the assignment $x_1 = \texttt{true}, x_2 = \texttt{true}, x_3 = \texttt{false}$. In the first snapshot (a), the superscripts of the vertices used in the proof of Theorem 4.1 are shown. To keep the figure clean, the superscripts are omitted in the illustrations for snapshots (b) and (c).

extra color for Minimum SW-Temporal Coloring parameterized by the vertex cover number of the underlying graph.

4.1 Refined NP-Hardness Results

We now present the main computational hardness result of this section. In particular, we show that SW-Temporal Coloring is NP-complete for $k = 2$ even if the input temporal graph has lifetime three.

Theorem 4.1. SW-Temporal Coloring *is NP-complete for all* $k \geq 2$, $\Delta \geq 2$, *and* $T \geq \Delta + 1$, *even if*

- *the underlying graph is* $(k+1)$*-colorable,*
- *the underlying graph has a maximum degree in* $O(k)$*, and*
- *every snapshot has connected components with size in* $O(k)$*.*

Proof. We present a reduction from Exact $(3,4)$-SAT [44] to SW-Temporal Coloring with $k = 2$ and $\Delta = 2$. The reduction can be easily modified to a larger number of colors, we explain how to do this at the end of the proof. Recall that in Exact $(3,4)$-SAT we are asked to decide whether a given Boolean formula ϕ is satisfiable and ϕ is in conjunctive normal form where every clause has exactly three distinct literals and every variable appears in exactly four clauses.

On an intuitive level, the main idea that we exploit in this reduction is that no matter how a triangle is colored with two colors, always (exaclty) one of the three edges is monochromatic. We use this idea both to construct variable gadgets (by further enforcing that a specific edge of the triangle always has to be properly colored) and to construct clause gadgets, where the three edges of a triangle correspond to the three literals in a clause and the monochromatic edge "selects" which literal should satisfy the clause.

Given a formula ϕ with n variables and m clauses, we construct a temporal graph $\mathcal{G} = (G, \lambda)$ consisting of $T = 3$ snapshots, which we will refer to as $G_1 = (V, E_1)$, $G_2 = (V, E_2)$, and $G_3 = (V, E_3)$. We construct the following variable gadgets and clause gadgets. An illustration of the construction is given in Figure 4.

Variable gadget. For each variable x_i with $i \in [n]$ of ϕ we create five vertices $v_{x_i}^{(1)}$, $v_{x_i}^{(2)}$, $v_{x_i}^{(3)}$, $v_{x_i}^{(4)}$, and $v_{x_i}^{(5)}$. The vertices $v_{x_i}^{(1)}$, $v_{x_i}^{(2)}$, and $v_{x_i}^{(3)}$ form a (not necessarily induced) P_3 in every snapshot, that is $\{v_{x_i}^{(1)}, v_{x_i}^{(2)}\} \in E_t$ and $\{v_{x_i}^{(2)}, v_{x_i}^{(3)}\} \in E_t$ for all $t \in [3]$. Furthermore, we connect $v_{x_i}^{(1)}$ and $v_{x_i}^{(3)}$ in the second

snapshot, that is, $\{v_{x_i}^{(1)}, v_{x_i}^{(3)}\} \in E_2$. Lastly, we create a full C_5 in snapshot three, that is, $\{v_{x_i}^{(3)}, v_{x_i}^{(4)}\} \in E_3$, $\{v_{x_i}^{(4)}, v_{x_i}^{(5)}\} \in E_3$, and $\{v_{x_i}^{(1)}, v_{x_i}^{(5)}\} \in E_3$.

Clause gadget. For each clause c_i with $i \in [m]$ of ϕ we create a total of 18 vertices. We create vertices $v_{c_i}^{(1)}$, $v_{c_i}^{(2)}$, and $v_{c_i}^{(3)}$ and connect them to form a triangle in every snapshot, that is, $\{v_{c_i}^{(1)}, v_{c_i}^{(2)}\} \in E_t$, $\{v_{c_i}^{(2)}, v_{c_i}^{(3)}\} \in E_t$, and $\{v_{c_i}^{(1)}, v_{c_i}^{(3)}\} \in E_t$ for all $t \in [3]$. In this proof, we refer to these vertices as the *core* of the clause gadget related to clause c_i. Next, we add six vertices, which we refer to as the *extension* of the core of the clause gadget related to clause c_i. Let these vertices be called $v_{c_i}^{(1,1)}$, $v_{c_i}^{(1,2)}$, $v_{c_i}^{(2,1)}$, $v_{c_i}^{(2,2)}$, $v_{c_i}^{(3,1)}$, and $v_{c_i}^{(3,2)}$. We connect $v_{c_i}^{(j,1)}$ and $v_{c_i}^{(j,2)}$ for all $j \in [3]$ in every snapshot, that is, $\{v_{c_i}^{(j,1)}, v_{c_i}^{(j,2)}\} \in E_t$ for all $j \in [3]$ and for all $t \in [3]$. In the second snapshot, we connect the extension and the core in the following way.

- Edge $\{v_{c_i}^{(1,1)}, v_{c_i}^{(1,2)}\}$ forms a C_4 with edge $\{v_{c_i}^{(2)}, v_{c_i}^{(1)}\}$, that is, $\{v_{c_i}^{(2)}, v_{c_i}^{(1,2)}\} \in E_2$ and $\{v_{c_i}^{(1)}, v_{c_i}^{(1,1)}\} \in E_2$.
- Edge $\{v_{c_i}^{(2,1)}, v_{c_i}^{(2,2)}\}$ forms a C_4 with edge $\{v_{c_i}^{(2)}, v_{c_i}^{(3)}\}$, that is, $\{v_{c_i}^{(2)}, v_{c_i}^{(2,1)}\} \in E_2$ and $\{v_{c_i}^{(3)}, v_{c_i}^{(2,2)}\} \in E_2$.
- Edge $\{v_{c_i}^{(3,1)}, v_{c_i}^{(3,2)}\}$ forms a C_4 with edge $\{v_{c_i}^{(1)}, v_{c_i}^{(3)}\}$, that is, $\{v_{c_i}^{(1)}, v_{c_i}^{(3,2)}\} \in E_2$ and $\{v_{c_i}^{(3)}, v_{c_i}^{(3,1)}\} \in E_2$.

Lastly, we introduce nine auxiliary vertices that help to connect clause gadgets and variable gadgets. Let these vertices be called $v_{c_i}^{(j,1,1)}$, $v_{c_i}^{(j,1,2)}$, and $v_{c_i}^{(j,2,1)}$ for all $j \in [3]$. In the third snapshot, we connect the extension of the core and these auxiliary vertices in the following way. For all $j \in [3]$ we have that $\{v_{c_i}^{(j,1,1)}, v_{c_i}^{(j,1,2)}\} \in E_3$, $\{v_{c_i}^{(j,1,2)}, v_{c_i}^{(j,1)}\} \in E_3$, and $\{v_{c_i}^{(j,2,1)}, v_{c_i}^{(j,2)}\} \in E_3$.

Connection of variable and clause gadgets. The clause gadgets and variable gadgets are connected in the third snapshot. Let clause $c_i = (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$ with $i \in [m]$ have literals $\ell_{i,1}$, $\ell_{i,2}$, and $\ell_{i,3}$. Let $x_{i,j}$ with $i \in [m]$ and $j \in [3]$ be the variable of the jth literal in clause c_i. If $\ell_{i,j} = x_{i,j}$, then $\{v_{x_{i,j}}^{(2)}, v_{c_i}^{(j,1,1)}\} \in E_3$ and $\{v_{x_{i,j}}^{(3)}, v_{c_i}^{(j,2,1)}\} \in E_3$. If $\ell_{i,j} = \neg x_{i,j}$, then $\{v_{x_{i,j}}^{(1)}, v_{c_i}^{(j,1,1)}\} \in E_3$ and $\{v_{x_{i,j}}^{(2)}, v_{c_i}^{(j,2,1)}\} \in E_3$. This completes the construction. Recall that $\Delta = 2$ and $k = 2$.

Properties of $\mathcal{G}$. We can check that the underlying graph is 3-colorable: It is easy to see that we can color each variable gadget with three colors in the underlying graph. The same for each clause gadget (without the connecting auxiliary vertices). The auxiliary vertices can now be colored as follows. Vertices $v_{c_i}^{(j,2,1)}$ are connected to two vertices with potentially different colors. Hence, we can use the third color for $v_{c_i}^{(j,2,1)}$. We can color $v_{c_i}^{(j,1,2)}$ with a color that is different from the color of $v_{c_i}^{(j,1)}$. Now $v_{c_i}^{(j,1,1)}$ is connected to two vertices with potentially different colors. Hence, we can use the third color for $v_{c_i}^{(j,1,1)}$.

To see that the underlying graph has constant maximum degree recall that every variable appears in exactly four clauses. Hence, the vertices $v_{x_i}^{(1)}$ have degree at most seven in the underlying graph. It is straightforward to check that all other vertices also have degree at most seven.

Lastly, we can easily verify that all snapshots are composed of small connected components (see also Figure 4). To see this, recall that every variable appears in exacty four clauses, hence in the third snapshot, each variable gadget is connected to four extensions of clause gadgets.

Correctness. It is easy to check that the reduction can be computed in polynomial time. It remains to show that $\mathcal{G}$ admits a proper sliding 2-window temporal 2-coloring if and only if ϕ is satisfiable.

($\Rightarrow$): Assume that we are given a satisfying assignment for ϕ. Then we construct a proper sliding 2-window temporal 2-coloring for $\mathcal{G}$ as follows. We start coloring the second snapshot and then show that we can color snapshots one and three in a way such that the complete coloring is a proper sliding 2-window temporal 2-coloring. If a variable x_i with $i \in [n]$ is set to true in the satisfying assignment, then we color the triangle of the corresponding variable gadget in a way that leaves only edge $\{v_{x_i}^{(1)}, v_{x_i}^{(2)}\}$ monochromatic. To be specific, assume (for the remainder of this paragraph) we have colors yellow and blue, we color vertices $v_{x_i}^{(1)}$ and $v_{x_i}^{(2)}$ in yellow and vertices $v_{x_i}^{(3)}$, $v_{x_i}^{(4)}$, and $v_{x_i}^{(5)}$ in blue. If variable x_i is set to false in the satisfying assignment, then we color the triangle of the corresponding variable gadget in a way that leaves edge $\{v_{x_i}^{(2)}, v_{x_i}^{(3)}\}$ monochromatic. To be specific, we color vertices $v_{x_i}^{(2)}$ and $v_{x_i}^{(3)}$ in yellow and vertices $v_{x_i}^{(1)}$, $v_{x_i}^{(4)}$, and $v_{x_i}^{(5)}$ in blue. For each clause c_i with $i \in [m]$ we choose one of its

literals that satisfies the clause. Let the jth literal with $j \in [3]$ be a satisfying literal of clause c_i for the given assignment. Then we color the core of the corresponding clause gadget in a way that leaves edge $\{v_{c_i}^{(j)}, v_{c_i}^{(j \bmod 3+1)}\}$ monochromatic. Note that coloring the core uniquely determines how we have to color the extension of the core since the connecting edges are only present in the third snapshot and hence have to be properly colored. The auxiliary vertices can be colored arbitrarily.

Now we show how to color snapshot one. For each variable x_i with $i \in [n]$, we color $v_{x_i}^{(2)}$ in yellow and the remaining vertices of the corresponding gadget in blue. Note that this ensures that the edge which remains monochromatic in the second snapshot is properly colored in the first snapshot. For each clause c_i with $i \in [m]$ we color the core in a way that ensures that the edge which remains monochromatic in the second snapshot is properly colored in the first snapshot. We properly color all edges of the extension and the auxiliary vertices arbitrarily. It is not hard to see that now the first Δ-window is properly colored.

Lastly, we show how to color the third snapshot. Note that for the variable gadgets, the coloring in snapshot two determines (up to renaming the colors) how to color the variable gadgets in the third snapshot. This also determines how to color the auxiliary vertices and the extension of the core in the third snapshot. This potentially leaves edges of the extension monochromatic. Note that in the second snapshot, all extension edges are properly colored except the one which, in the third snapshot, is connected to a variable that, in the given assignment, satisfies the clause. It is straightforward to check that in this case, this particular extension edge is properly colored in the third snapshot. Lastly, the core is colored in a way that ensures that the edge that is colored monochromatically in the second snapshot is colored properly in the third snapshot. It is easy to check that now the second Δ-window is also properly colored.

($\Leftarrow$): Assume we are given a proper sliding 2-window temporal 2-coloring for $\mathcal{G}$. Then we construct a satisfying assignment for ϕ in the following way: Note that in the second snapshot each variable gadget contains a triangle with exactly one monochromatic edge. The edge $\{v_{x_i}^{(1)}, v_{x_i}^{(3)}\}$ only exists in the second snapshot and hence is colored properly by any proper sliding 2-window temporal 2-coloring. This means that either edge $\{v_{x_i}^{(1)}, v_{x_i}^{(2)}\}$ or edge $\{v_{x_i}^{(2)}, v_{x_i}^{(3)}\}$ is colored monochromatic. If $\{v_{x_i}^{(1)}, v_{x_i}^{(2)}\}$ is colored monochromatically, then we set x_i to true, otherwise we set x_i to false. We claim that this yields a satisfying assignment for ϕ.

Assume for contradiction that it is not. Then there is a clause c_j that is not satisfied. Without loss of generality, let x_1, x_2, and x_3 be the variables appearing in c_j. Then in the third snapshot, the clause gadget of c_j is connected to the variable gadgets of x_1, x_2, and x_3. It is easy to check that in any proper sliding 2-window temporal 2-coloring, exactly one edge of the extension of any clause gadget is colored monochromatic in the second snapshot, hence this is also the case in the clause gadget of c_j. Without loss of generality, let the monochromatically colored (in the second snapshot) extension edge of the clause gadget of c_j be connected to the variable gadget of x_1 in the third snapshot. It is easy to check that for the sliding 2-window temporal 2-coloring to be proper, the edge of the variable gadget of x_1 that is connected to the clause gadget of c_j in the third snapshot needs to be colored properly in the second snapshot. By construction of $\mathcal{G}$ this is a contradiction to c_j not being satisfied by the constructed assignment.

Modification for a Larger Number of Colors. To modify this reduction for more colors we introduce new vertices and edges to all snapshots to "block" all colors except two from being used. Formally, we do the following. Let $k > 2$. For each snapshot $i \in [3]$, we add $k-2$ fresh vertices $c_1^{(i)}, \ldots, c_{k-2}^{(i)}$ for each connected component C in that snapshot. The vertices $c_1^{(i)}, \ldots, c_{k-2}^{(i)}$ form a clique in snapshot i, and we connect them to all vertices in the connected component C. In all snapshots different from i the vertices $c_1^{(i)}, \ldots, c_{k-2}^{(i)}$ are isolated. All new edges exist in exactly one snapshot and hence have to be colored properly in this snapshot. It follows that the vertices $c_1^{(i)}, \ldots, c_{k-1}^{(i)}$ have to be colored with $k-2$ distinct colors and these colors cannot be used to color any other vertex from the connected component in snapshot i.

The number of new vertices introduced by this modification is in $O(n \cdot k)$. It is easy to check that this increases the number of colors necessary to color the underlying graph by $k-2$. The maximum degree of the underlying graph after the modification is in $O(k)$ and the size of each connected component in each snapshot is increased by $k-2$. □

With small modifications to the reduction we get that SW-Temporal Coloring remains hard

under the following restrictions on the snapshots.

Corollary 4.2. SW-Temporal Coloring *is NP-complete for all* $k \geq 2$*,*

- $\Delta \in O(k^2)$*, and* $T \geq \Delta + 1$ *even if every snapshot is a cluster graph (i.e. the disjoint union of complete graphs), or*
- $\Delta \geq 3$*, and* $T \geq \Delta + 1$ *even if every snapshot has domination number one.*

Proof. Both modifications rely on the following construction. We can insert additional vertices and edges to each of the three snapshots of the reduction presented in the proof of Theorem 4.1. Between snapshots two and three we add sufficiently many new snapshots containing exclusively new edges, such that all new edges can be properly temporally colored at least once if Δ is increased by the number of new snapshots. Now all new edges can be colored properly in the newly inserted snapshots and the original construction of the reduction is not affected.

To get the first property for all snapshots, we can add all edges that transform each connected component into a clique to the three original snapshots. Since the components have size $O(k)$, we only add a number of new edges that is in $O(k^2)$ per component. Hence, we can add a number of new snapshots in $O(k^2)$ each containing one new edge per component. The newly added snapshots are clearly cluster graphs, hence we get the result.

To get the second property, we add one new universal vertex and one new snapshot that contains all new edges. Clearly, now all snapshots have a dominating set of size one. □

We remark that Theorem 4.1 and Corollary 4.2 have interesting implications from a parameterized point of view. Parameterizing SW-Temporal Coloring by the maximum degree of the underlying graph cannot yield fixed-parameter tractability unless P = NP, even if combined with k and T. Furthermore, parameterizing SW-Temporal Coloring by structural graph parameters of the snapshots that are constant if all connected components are constant-sized cannot yield fixed-parameter tractability unless P = NP, even if combined with k and T. Parameters for which this holds are for example "treedepth" and all structurally smaller parameters (including for example "treewidth"). The same holds for structural graph parameters of the snapshots that are constant for cluster graphs, such as for example "vertex deletion distance to cluster graph".

The reduction presented in the proof of Theorem 4.1 also yields a running time lower bound assuming the Exponential Time Hypothesis (ETH) [30, 31]. In the next section we will use this result to show that an algorithm we present presumably is asymptotically optimal if Δ is constant.

Corollary 4.3. SW-Temporal Coloring *does not admit an* $f(k+T)^{o(|\mathcal{G}|)} \cdot |\mathcal{G}|^{f(k+T)}$*-time algorithm for any computable function* f *unless the ETH fails.*

Proof. First, note that any 3-SAT formula with m clauses can be transformed into an equisatisfiable Exact $(3,4)$-SAT formula with $O(m)$ clauses [44]. The reduction presented in the proof of Theorem 4.1 produces an instance of SW-Temporal Coloring with a temporal graph of size $|\mathcal{G}| \in O(m)$, $k = 2$, and $T = 3$. Hence an algorithm for SW-Temporal Coloring with running time $f(k+T)^{o(|\mathcal{G}|)} \cdot |\mathcal{G}|^{f(k+T)}$ for some computable function f would imply the existence of an $2^{o(m)}$-time algorithm for 3-SAT. This is a contradiction to the ETH [30, 31]. □

4.2 An Exponential-Time Algorithm for Sliding Window Temporal Coloring

In the following we give an exponential-time algorithm for SW-Temporal Coloring that, if Δ is constant, asymptotically matches the running time lower bound given in Corollary 4.3 assuming the ETH to hold.

We start with an auxiliary technical observation that, intuitively, allows us to combine partial colorings if they agree on their overlap and the overlap is sufficiently large. The observation is illustrated in Figure 5.

Observation 4.4. *Let* $(\mathcal{G}_1 = (V, E_1, E_2, \ldots, E_i), k, \Delta)$ *and* $(\mathcal{G}_2 = (V, E_j, E_{j+1}, \ldots, E_T), k, \Delta)$ *be two instances of* SW-Temporal Coloring *with* $j + \Delta - 1 \leq i \leq T$*. Let* $\Upsilon^{(1)}$ *and* $\Upsilon^{(2)}$ *be sliding* Δ*-window temporal colorings for* $\mathcal{G}_1$ *and* $\mathcal{G}_2$*, respectively, that use the same* k *colors and with the property that for all* $v \in V$ *and for all* $j \leq i^\star \leq i$ *we have that* $\Upsilon^{(1)}(v, i^\star) = \Upsilon^{(2)}(v, i^\star)$*.*

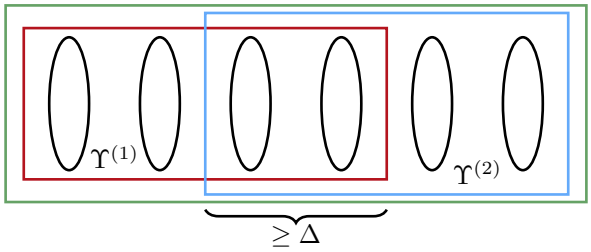


Figure 5: Illustration of Observation 4.4. The ellipses visualize the snapshots of the temporal graphs. The red and blue boxes indicate the snapshots that are colored by the proper sliding Δ-window temporal colorings $\Upsilon^{(1)}$ and $\Upsilon^{(2)}$, respectively. By Observation 4.4, if $\Upsilon^{(1)}$ and $\Upsilon^{(2)}$ overlap on at least Δ snapshots, which are colored in the same way in both $\Upsilon^{(1)}$ and $\Upsilon^{(2)}$, then the combination of both colorings $\Upsilon^{(1)}$ and $\Upsilon^{(2)}$ (illustrated by the green box) is also a proper sliding Δ-window temporal coloring.

Then we have that $\Upsilon^{(1)}$ and $\Upsilon^{(2)}$ are proper sliding Δ-window temporal colorings if and only if $(\Upsilon_1^{(1)}, \Upsilon_2^{(1)}, \ldots, \Upsilon_i^{(1)}, \Upsilon_{i+1}^{(2)}, \ldots, \Upsilon_T^{(2)})$ is a proper sliding Δ-window temporal coloring for $\mathcal{G} = (V, E_1, E_2, \ldots, E_T)$.

Proof. It is easy to see that if $(\Upsilon_1^{(1)}, \Upsilon_2^{(1)}, \ldots, \Upsilon_i^{(1)}, \Upsilon_{i+1}^{(2)}, \ldots, \Upsilon_T^{(2)})$ is a proper sliding Δ-window temporal coloring for $\mathcal{G} = (V, E_1, E_2, \ldots, E_T)$, then we have that $\Upsilon^{(1)}$ and $\Upsilon^{(2)}$ are proper sliding Δ-window temporal colorings for $(\mathcal{G}_1 = (V, E_1, E_2, \ldots, E_i), k, \Delta)$ and $(\mathcal{G}_2 = (V, E_j, E_{j+1}, \ldots, E_T), k, \Delta)$, respectively. For the other direction, assume for contradiction that $(\Upsilon_1^{(1)}, \Upsilon_2^{(1)}, \ldots, \Upsilon_i^{(1)}, \Upsilon_{i+1}^{(2)}, \ldots, \Upsilon_T^{(2)})$ is *not* a proper sliding Δ-window temporal coloring for $\mathcal{G} = (V, E_1, E_2, \ldots, E_T)$. Then there is a Δ-window W_t^Δ for some t and an edge $e \in E_{t'}$ for some $t' \in W_t^\Delta$ that is never properly colored in W_t^Δ. However, it is easy to check that W_t^Δ is completely contained in $[i]$ or $[j, T]$ (because of the bounds on i and j) and hence $\Upsilon^{(1)}$ or $\Upsilon^{(2)}$ color the whole Δ-window W_t^Δ. Since, by assumption, both $\Upsilon^{(1)}$ and $\Upsilon^{(2)}$ are proper sliding Δ-window temporal colorings, there being an edge that exists in W_t^Δ and is not properly colored is a contradiction. □

Now we are ready to describe an exponential-time algorithm for SW-Temporal Coloring. The main idea is to enumerate all partial proper sliding Δ-window temporal colorings for temporal subgraphs of lifetime 2Δ and then to check whether we can combine them to a proper sliding Δ-window temporal coloring for the whole temporal graph using Observation 4.4.

Theorem 4.5. SW-Temporal Coloring *can be solved in $O(k^{4\Delta \cdot |V|} \cdot T)$ time.*

Proof. Let $(\mathcal{G} = (G, \lambda), k, \Delta)$ be an input instance for SW-Temporal Coloring. For the sake of simplicity, we assume that T is divisible by Δ. If this is not the case, we can "mirror" the last snapshots: we repeat snapshots $\{T - (T \bmod \Delta) - 1, \ldots, T - 1\}$ in reverse order after the Tth snapshot. We give an algorithm for this problem that works as follows:

1. For 2Δ-windows $W_{i\Delta+1}^{2\Delta} = [i\Delta + 1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, enumerate all partial proper sliding Δ-window temporal colorings $\Upsilon_{W_{i\Delta+1}^{2\Delta}}^{(j)}$ that use at most k fixed colors, where each trivial snapshot is colored in some fixed but arbitrary way[2].
2. Create a *directed acyclic graph* (DAG) with all $\Upsilon_{W_{i\Delta+1}^{2\Delta}}^{(j)}$ as vertices and connect $\Upsilon_{W_{i\Delta+1}^{2\Delta}}^{(j)}$ and $\Upsilon_{W_{(i+1)\Delta+1}^{2\Delta}}^{(j')}$ with a directed arc if the two proper sliding Δ-window temporal colorings agree on the overlapping part.
3. Create a source vertex s and connect it to all $\Upsilon_{W_1^{2\Delta}}^{(j)}$ with a directed arc and create a sink vertex z and add a directed arc from all $\Upsilon_{W_{(T/\Delta-2)\Delta+1}^{2\Delta}}^{(j)}$ to it.

[2]This is an important trick that allows us to use this algorithm for the FPT result in Theorem 4.6.

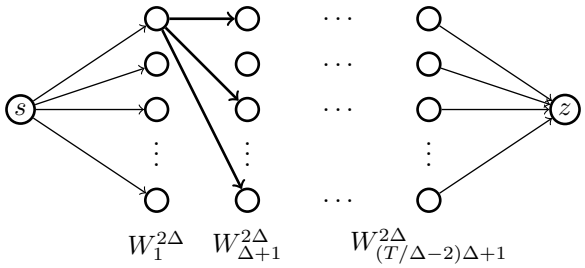


Figure 6: Illustration of the DAG constructed in the algorithm described in the proof of Theorem 4.5. Each vertex in each column of vertices corresponds to a partial proper sliding Δ-window temporal coloring for the 2Δ-window written at the bottom of the column. Thick arcs are only included in the DAG if the two connected partial proper sliding Δ-window temporal colorings agree on the overlapping part.

4. If there is a path from s to z, then answer YES, otherwise NO.

The constructed DAG is visualized in Figure 6.

Correctness. We now show that the above described algorithm is correct.

($\Rightarrow$): Assume that we are given a proper sliding Δ-window temporal coloring Υ that uses at most k colors for $\mathcal{G}$. Without loss of generality, let the k colors be the same fixed k colors we use in the algorithm. If $\mathcal{G}$ contains trivial snapshots, we assume without loss of generality that Υ colors them in the same fixed but arbitrary way as we do in the algorithm. Then for 2Δ-windows $W_{i\Delta+1}^{2\Delta} = [i\Delta+1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, the partial coloring of $W_{i\Delta+1}^{2\Delta}$ that agrees with Υ appears in the constructed DAG, since by assumption it is proper and we enumerate all of them. Now for any two 2Δ-windows $W_{i\Delta+1}^{2\Delta}$ and $W_{(i+1)\Delta+1}^{2\Delta}$ we obviously have that if we color them with Υ the overlapping part is colored in the same way. Hence the vertices corresponding to the implied partial coloring for these two 2Δ-windows are connected. Following these connections, we can see that we find a path from s to z in the constructed DAG.

($\Leftarrow$): If there is a path from s to z in the constructed DAG, then, by Observation 4.4, we can combine the partial proper sliding Δ-window temporal coloring corresponding to the vertices visited by the path since, by construction, they overlap for Δ time slots and agree on how to color the vertices in the overlapping part. This gives us a proper sliding Δ-window temporal coloring for the whole graph.

Running Time. The running time is dominated by checking whether s and z are connected in the last step of the algorithm. This can be done for example by a breadth-first-search on the constructed DAG. The DAG has at most $k^{2\Delta\cdot|V|} \cdot T$ vertices and at most $k^{4\Delta\cdot|V|} \cdot T$ edges. □

4.3 An FPT-Algorithm for Sliding Window Temporal Coloring

In this section, we show how to extend the algorithm presented in Theorem 4.5 to achieve linear time fixed-parameter tractability with respect to the number of vertices. The main idea is to reduce the number of non-trivial snapshots in each Δ-window. However, the procedure we describe only guarantees a very large upper bound on the number of non-trivial snapshots in each Δ-window. Hence, the following result is only of classification nature.

Theorem 4.6. SW-Temporal Coloring *can be solved in* $2^{O(2^{|V|^2})} \cdot T$ *time.*

Proof. We present a preprocessing step to reduce the number of non-trivial snapshots in any Δ-window and then use the algorithm of Theorem 4.5 to solve the problem.

The data reduction rule is based on the observation that if some snapshot appears at least $|V|^2$ times in a Δ-window, then the edges of this snapshot can be properly colored with two colors within the Δ-window. In other words, all but $|V|^2$ copies of the snapshot in the Δ-window are redundant for optimal coloring and each of them could be replaced by the trivial snapshot. When implementing this idea one should take care to guarantee that replacing a snapshot by the trivial one does not reduce the number of copies of the snapshot in other Δ-windows which contain at most $|V|^2$ copies of the snapshot.

Formally, the data reduction rule is as follows. Since the number of different snapshots is at most $2^{\binom{|V|}{2}} \leq 2^{|V|^2}$, by the pigeonhole principle if $\Delta > 2 \cdot 2^{|V|^2} \cdot |V|^2$, then in every Δ-window there exists a snapshot that appears more than $2|V|^2$ times in that Δ-window. For every such snapshot that contains at least one edge, we replace one of its "middle" copies, that is, one that has at least $|V|^2$ copies appearing earlier and $|V|^2$ copies that appear later in the Δ-window by a trivial snapshot. This data reduction rule guarantees that every Δ-window that contains the modified snapshot also contains at least $|V|^2$ copies of the original snapshot appearing either earlier or later in the Δ-window.

The data reduction rule can be applied exhaustively by linearly sweeping over all Δ-windows once in the following way. For each different graph (snapshot) we store a list of occurrences and update these lists every time we move the Δ-window by one. Having these lists, it is straightforward to count the occurrences and replace the middle ones by trivial snapshots. When we move the Δ-window, we just have to update two lists: the one of the graph that enters the Δ-window and the one of the graph that leaves. This requires a lookup table of size $2^{\binom{|V|}{2}} \leq 2^{|V|^2}$ but takes only time linear in T. Note that after this procedure, every Δ-window contains at most $2 \cdot 2^{|V|^2} \cdot |V|^2 \in 2^{O(2^{|V|^2})}$ non-trivial snapshots. Now we apply the algorithm of Theorem 4.5. Since we can assume that $k \leq |V|$, the number of colorings that are enumerated in Step 1 of the algorithm in Theorem 4.5 is in $2^{O(2^{|V|^2})}$. This the case because the number of enumerated colorings only depends on the number of non-trivial snapshots in each Δ-window. This completes the proof. □

We complement the fixed-parameter tractability result of Theorem 4.6 with the following proposition, in which we exclude the possibility of a polynomial-sized kernel for SW-Temporal Coloring when parameterized by the number $|V|$ of vertices unless NP $\subseteq$ coNP/poly. This stands in contrast to the existence of a polynomial kernel for Temporal Coloring when parameterized by $|V|$ [37].

Proposition 4.7. SW-Temporal Coloring *parameterized by the number $|V|$ of vertices does not admit a polynomial kernel for all $\Delta \geq 2$ and $k \geq 2$ unless NP $\subseteq$ coNP/poly.*

We need the following notation for the proof. An equivalence relation R on the instances of some problem L is a *polynomial equivalence relation* if

(i) one can decide for each two instances in time polynomial in their sizes whether they belong to the same equivalence class, and
(ii) for each finite set S of instances, R partitions the set into at most $(\max_{x \in S} |x|)^{O(1)}$ equivalence classes.

An *AND-cross-composition* of a problem $L \subseteq \Sigma^*$ into a parameterized problem P (with respect to a polynomial equivalence relation R on the instances of L) is an algorithm that takes n R-equivalent instances $x_1, \ldots, x_n$ of L and constructs in time polynomial in $\sum_{i=1}^{n} |x_i|$ an instance (x, k) of P such that

(i) k is polynomially upper-bounded in $\max_{1 \leq i \leq n} |x_i| + \log(n)$ and
(ii) (x, k) is a yes-instance of P if and only if x_i is a yes-instance of L for every $i \in [n]$.

If an NP-hard problem L AND-cross-composes into a parameterized problem P, then P does not admit a polynomial-size kernel, unless NP $\subseteq$ coNP/poly [7, 16], which would cause a collapse of the polynomial-time hierarchy to the third level.

Proof of Proposition 4.7. We provide an AND-cross-composition from Exact $(3, 4)$-SAT [44]. Recall that in Exact $(3, 4)$-SAT we are asked to decide whether a given Boolean formula ϕ is satisfiable and ϕ is in conjunctive normal form where every clause has exactly three distinct literals and every variable appears in exactly four clauses. Intuitively, we can just string together instances produced by the reduction we presented in the proof of Theorem 4.1 in the time axis with some extra snapshots in between such that the large instance admits a proper sliding Δ-window temporal coloring if and only if all original instances are YES-instances.

We define an equivalence relation R as follows: Two instances ϕ and ψ are equivalent under R if and only if the number of variables and the number of clauses is the same in both formulas. Clearly, R is a polynomial equivalence relation.

Now let $\phi_1, \ldots, \phi_n$ be R-equivalent instances of Exact $(3, 4)$-SAT. We arbitrarily number all variables and clauses of all formulas. For each ϕ_i with $i \in [n]$ we construct an instance of SW-Temporal Coloring as defined in the proof of Theorem 4.1 (for an illustration see Figure 4) with the

only difference that we add a fourth and fifth snapshot both of which are copies of the first snapshot (Figure 4a). Now we put all constructed temporal graphs next to each other in temporal order, that is, if $\mathcal{G}^{(i)} = (V, E_1^{(i)}, E_2^{(i)}, \ldots, E_5^{(i)})$ is the graph constructed for ϕ_i, then the overall temporal graph is $\mathcal{G} = (V, E_1^{(1)}, E_2^{(1)}, \ldots, E_5^{(1)}, E_1^{(2)}, E_2^{(2)}, \ldots, E_5^{(2)}, \ldots, E_1^{(n)}, E_2^{(n)}, \ldots, E_5^{(n)})$. Here, the vertex set stays the same. We identify the vertices with their names according to the numbering of the variables and clauses of the formulas. Further, we set $\Delta = 2$ and $k = 2$.

This instance can be constructed in polynomial time and the number of vertices is linearly upper-bounded in the size of the formulas, hence $|V|$ is polynomially upper-bounded by the maximum size of an input instance. Furthermore, it is easy to check that the two extra copies of the first snapshot in the construction (Figure 4a) allow to go from an arbitrary proper coloring of snapshot $G_4^{(i)} = (V, E_4^{(i)})$ to $G_1^{(i+1)} = (V, E_1^{(i+1)})$ for any $i \in [n-1]$. It follows from the proof of Theorem 4.1, that the constructed SW-Temporal Coloring instance is a YES-instance if and only if for every $i \in [n]$ formula ϕ_i is satisfiable.

Since Exact $(3, 4)$-SAT is NP-hard [44] and we AND-cross-composed it into SW-Temporal Coloring with $\Delta = 2$ and $k = 2$ parameterized by $|V|$, the result follows. □

4.4 Structural Graph Parameters and Approximation

In this section, we investigate the possibility to improve the fixed-parameter tractability result of Theorem 4.6 by replacing the parameter $|V|$ with a smaller parameter. We answer this negatively by showing that SW-Temporal Coloring remains NP-complete even if the underlying graph has vertex cover number in $O(k)$, which is a fairly large structural parameter.

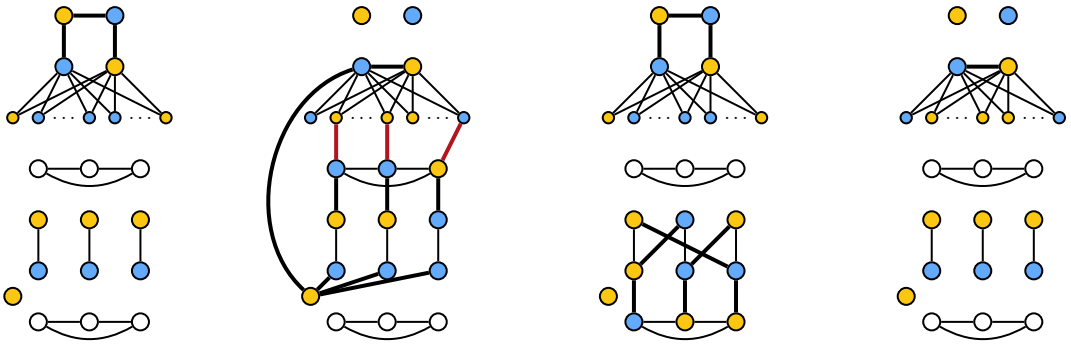
Theorem 4.8. SW-Temporal Coloring *is NP-complete for all $k \geq 2$, even if $\Delta = 2$ and the vertex cover number of the underlying graph is in $O(k)$.*

Proof. We present a reduction from Monotone Exactly 1-in-3 SAT [42, 23] to SW-Temporal Coloring with $k = 2$ and $\Delta = 2$. The reduction can be easily modified to a larger number of colors, we explain how to do this at the end of the proof. In Monotone Exactly 1-in-3 SAT we are given a collection of triples (clauses) of variables and the task is to determine whether there is an assignment of truth values to variables such that each clause contains exactly one variable that is set to true. Given an instance I of Monotone Exactly 1-in-3 SAT with n variables and m clauses, we construct a temporal graph $\mathcal{G} = (G, \lambda)$ with $T = 4m$ snapshots in the following way. The construction is visualized in Figure 7.

Construction. In the construction, we classify the snapshots of the constructed temporal graph by the remainders of their time slots when divided by four. This gives us *type 1*, *type 2*, *type 3*, and *type 4* snapshots, where type 4 snapshots are the ones with a time slot that is divisible by four and the other type numbers correspond to the remainders of the time slots. We start by adding four vertices u_1, u_2, u_3, and u_4 to G. In snapshots of type 1 or type 3 we add edges $\{u_1, u_2\}$, $\{u_1, u_3\}$, and $\{u_2, u_4\}$. In snapshots of type 2 or type 4 we add edge $\{u_3, u_4\}$. For each variable x_i we add a vertex v_i. We connect each of u_3 and u_4 to all v_i in all snapshots. Next, we add 13 further vertices $w_1, w_2, \ldots, w_{13}$ to V. In all snapshots we pairwise connect w_1, w_2, and w_3, pairwise connect w_{11}, w_{12}, and w_{13}, and add edges $\{w_4, w_7\}$, $\{w_5, w_8\}$, and $\{w_6, w_9\}$. In snapshots of type 2 we add edges $\{w_1, w_4\}$, $\{w_2, w_5\}$, $\{w_3, w_6\}$, $\{u_3, w_{10}\}$, $\{w_7, w_{10}\}$, $\{w_8, w_{10}\}$, and $\{w_9, w_{10}\}$ (see Figure 7b). In snapshots of type 3 we add edges $\{w_4, w_9\}$, $\{w_5, w_7\}$, $\{w_6, w_8\}$, $\{w_7, w_{11}\}$, $\{w_8, w_{12}\}$, and $\{w_9, w_{13}\}$ (see Figure 7c). Lastly, let x_{i_1}, x_{i_2}, and x_{i_3} be the three variables contained in clause c_j. Then we add edges $\{v_{i_1}, w_1\}$, $\{v_{i_2}, w_2\}$, and $\{v_{i_3}, w_3\}$ in snapshot $4j - 2$ (see red edges in Figure 7b). Note that snapshot $4j - 2$ has type 2.

Correctness. It is easy to check that this can be done in polynomial time. Note that vertices $u_1, u_2, u_3, u_4, w_1, w_2, \ldots, w_{13}$ form a vertex cover in G. We are ready now to prove that the Monotone Exactly 1-in-3 SAT instance I is a YES-instance if and only if $\mathcal{G}$ admits a proper sliding 2-window temporal 2-coloring.

($\Rightarrow$): Assume we are given a YES-instance of Monotone Exactly 1-in-3 SAT with a satisfying assignment. We show that the constructed instance of SW-Temporal Coloring is also a YES-instance by presenting a proper sliding Δ-window temporal coloring with two colors. Let blue and yellow be the two colors we use. We always color u_1 and u_4 yellow and u_2 and u_3 blue. If variable x_i is set to true



(a) Type 1 snapshot. (b) Type 2 snapshot. (c) Type 3 snapshot. (d) Type 4 snapshot.

Figure 7: Illustration of the reduction from Monotone Exactly 1-in-3 SAT to SW-Temporal Coloring of the proof of Theorem 4.8. The vertex numbering in the description of the construction corresponds to a row-wise numbering from top-left to bottom-right. The first two rows correspond to vertices u_1 to u_4. The third row corresponds to vertices v_1 to v_n. The remaining rows correspond to vertices w_1 to w_{13}. Thin edges appear in all snapshots. Thick edges never appear consecutively and hence need to be colored properly. Red edges correspond to clauses. The colors of the vertices correspond to the proper sliding Δ-window temporal coloring constructed in the proof of Theorem 4.8.

in the satisfying assignment, then we color v_i yellow in snapshots of type 1 and 3 and blue in snapshots of type 2 and 4. If variable x_i is set to false in the satisfying assignment, then we color v_i yellow in snapshots of type 2 and 4 and blue in snapshots of type 1 and 3. Vertex w_{10} is always colored yellow.

Let clause c_j be satisfied by its sth variable (note that $s \in [3]$). We describe how to color vertices $w_1, \ldots, w_9$ in snapshot $4 \cdot j - 2$. Vertex w_s is colored yellow. Vertices in $\{w_1, w_2, w_3\} \setminus \{w_s\}$ are colored blue. Vertex w_{s+3} is colored blue. Vertices in $\{w_4, w_5, w_6\} \setminus \{w_{s+3}\}$ are colored yellow. Vertices w_7, w_8, and w_9 are colored blue. We further describe how to color vertices $w_4, \ldots, w_{13}$ in snapshot $4 \cdot j - 1$. Note that w_{10} is already colored yellow.

- We color vertex $w_{((s+1) \bmod 3)+4}$ blue and vertices in $\{w_4, w_5, w_6\} \setminus \{w_{((s+1) \bmod 3)+4}\}$ yellow.
- We color vertex $w_{(s \bmod 3)+7}$ yellow and vertices in $\{w_7, w_8, w_9\} \setminus \{w_{(s \bmod 3)+7}\}$ blue.
- We color vertex $w_{(s \bmod 3)+11}$ blue and vertices in $\{w_{11}, w_{12}, w_{13}\} \setminus \{w_{(s \bmod 3)+11}\}$ yellow.

In all snapshots of type 1 and 4 we color vertices w_4, w_5, and w_6 yellow and vertices w_7, w_8, and w_9 blue. The coloring scheme so far is depicted in Figure 7. Note that the colors of some vertices in some snapshots are not specified yet. These are the white vertices in Figure 7. All these vertices belong to triangles, hence we can color them in a way that each triangle has one monochromatic edge. We choose as the edge that should remain monochromatic an edge that is properly colored in both adjacent snapshots. Such an edge always exists since all these triangles are also triangles in the adjacent snapshots and a triangle is never colored completely monochromatic.

($\Leftarrow$): Assume that the constructed instance of SW-Temporal Coloring is a YES-instance and that we have a proper sliding Δ-window temporal coloring with two colors. We show that the given instance of Monotone Exactly 1-in-3 SAT is also a YES-instance by constructing a satisfying assignment. We claim that the following yields a satisfying assignment. For every variable x_i, if edge $\{u_3, v_i\}$ is colored properly in the first snapshot, then we set x_i to true, otherwise we set x_i to false.

First we argue that if an edge $\{u_3, v_i\}$ is colored properly in the first snapshot for some $i \in [n]$, then it is also colored properly in every odd snapshot (that is, every snapshot of type 1 and 3). Furthermore, the edge is colored monochromatically in every even snapshot (that is, every snapshot of type 2 and 4). Analogously, if an edge $\{u_3, v_i\}$ is colored monochromatically in the first snapshot for some $i \in [n]$, then it is also colored monochromatically in every odd snapshot and colored properly in every even snapshot. This follows from an easily verifiable fact that in every proper sliding Δ-window temporal coloring vertex u_3 is colored different from vertex u_4 in every snapshot. It follows that if an edge $\{u_3, v_i\}$

is colored monochromatically in a snapshot t for some $i \in [n]$ and $t \in [T-1]$, then $\{u_3, v_i\}$ needs to be colored properly in snapshot $t+1$ meaning that $\{u_4, v_i\}$ is colored monochromatically in snapshot $t+1$. Symmetrically, if an edge $\{u_4, v_i\}$ is colored monochromatically in a snapshot t for some $i \in [n]$ and $t \in [T-1]$, then $\{u_4, v_i\}$ needs to be colored properly in snapshot $t+1$ meaning that $\{u_3, v_i\}$ is colored monochromatically in snapshot $t+1$.

Now we are ready to argue that each clause of the Monotone Exactly 1-in-3 SAT instance is satisfied. To see this we first take a look at snapshots of type 3. Note that the triangle consisting of vertices w_{11}, w_{12}, and w_{13} has exactly one monochromatic edge. It cannot have three since not all three edges can be colored properly in the adjacent snapshots. This means that exactly two out of the three vertices w_{11}, w_{12}, and w_{13} have the same color. It follows that exactly two out of the three vertices w_7, w_8, and w_9 have the same color, since the edges $\{w_7, w_{11}\}$, $\{w_8, w_{12}\}$, and $\{w_9, w_{13}\}$ need to be colored properly. It is easy to check that this implies that exactly two out of the three edges $\{w_4, w_7\}$, $\{w_5, w_8\}$, and $\{w_6, w_9\}$ are colored monochromatically, since edges $\{w_4, w_9\}$, $\{w_5, w_7\}$, and $\{w_6, w_8\}$ need to be colored properly.

Now we take a look at snapshots of type 2. From the last paragraph follows that at most one out of the three edges $\{w_4, w_7\}$, $\{w_5, w_8\}$, and $\{w_6, w_9\}$ is colored monochromatically. Since edges $\{u_3, w_{10}\}$, $\{w_7, w_{10}\}$, $\{w_8, w_{10}\}$, and $\{w_9, w_{10}\}$ need to be colored properly, we have that vertices w_7, w_8, and w_9 have the same color as vertex u_3. It follows that at most one of the vertices w_4, w_5, and w_6 is colored in the same color as u_3. Since vertices w_1, w_2, and w_3 form a triangle and edges $\{w_1, w_4\}$, $\{w_2, w_5\}$, and $\{w_3, w_6\}$ need to be colored properly, it follows that exactly one out of the three vertices w_1, w_2, and w_3 is colored differently from u_3. Recall that w_1, w_2, and w_3 are connected to vertices v_{i_1}, v_{i_2}, and v_{i_3} corresponding to the three variables x_{i_1}, x_{i_2}, and x_{i_3} that are contained in the clause that corresponds to the snapshot. The connecting edges need to be colored properly. Consequently, exactly one of the edges $\{u_3, v_{i_1}\}$, $\{u_3, v_{i_2}\}$, and $\{u_3, v_{i_3}\}$ is colored monochromatically and the other two are colored properly. It follows that in the first snapshot, exactly one of the edges $\{u_3, v_{i_1}\}$, $\{u_3, v_{i_2}\}$, and $\{u_3, v_{i_3}\}$ is colored properly and the other two are colored monochromatically. This means we set exactly one of the three variables to true and the clause is satisfied.

Modification for a Larger Number of Colors. To modify this reduction for more colors we introduce new vertices and edges to the snapshots to "block" all colors except two from being used. Formally, we do the following. Let $k > 2$. We add $2k-4$ fresh vertices $c_1^{(1)}, \ldots, c_{k-2}^{(1)}$ and $c_1^{(2)}, \ldots, c_{k-2}^{(2)}$. In each snapshot of type 1 or 3 the vertices $c_1^{(1)}, \ldots, c_{k-2}^{(1)}$ form a clique and we connect them to all other vertices. In each snapshot of type 2 or 4 the vertices $c_1^{(2)}, \ldots, c_{k-2}^{(2)}$ form a clique and connect them to all other vertices. All new edges exist exactly once during any Δ-window for $\Delta = 2$ and hence have to be colored properly in every snapshot in which they appear. It follows that all vertices $c_1^{(i)}, \ldots, c_{k-1}^{(i)}$ have to be colored with $k-2$ distinct colors and these colors cannot be used to color any other vertex.

The number of new vertices introduced by this modification is in $O(k)$ and we can simply add all of them to the vertex cover of the underlying graph. □

Finally, we consider a canonical optimization version of SW-Temporal Coloring, which we call Minimum SW-Temporal Coloring, where the goal is to minimize the number of colors k. Using Theorem 4.6, we provide an FPT-approximation algorithm with an additive error of one where the parameter is the vertex cover number of the underlying graph. Considering that we cannot hope for an exact FPT algorithm for parameter "vertex cover number of the underlying graph" unless P = NP (Theorem 4.8), this is the best we can get from a classification standpoint.

Theorem 4.9. Minimum SW-Temporal Coloring *admits an approximation algorithm with a running time in* $2^{O(2^{\mathrm{vc}_\downarrow^2})} \cdot T$ *and an additive error of one, where* $\mathrm{vc}_\downarrow$ *is the vertex cover number of the underlying graph.*

Proof. Let $\mathcal{G} = (G, \lambda)$ be the input temporal graph. First, we compute a minimum vertex cover $S \subseteq V$ of the underlying graph G. Let the size of this vertex cover be $\mathrm{vc}_\downarrow = |S|$. Note that this can be done in $O(2^{\mathrm{vc}_\downarrow} \cdot (|V| + |E(G)|)$ time [9, 16]. We use the algorithm of Theorem 4.6 to compute the size of a minimum proper sliding Δ-window temporal coloring for the temporal graph $\mathcal{G}[S]$ induced by the vertex cover vertices[3]. By Theorem 4.6 this computation takes $2^{O(2^{\mathrm{vc}_\downarrow^2})} \cdot T$ time and the number of colors

[3]Note that the algorithm presented in Theorem 4.6 solves the decision version of SW-Temporal Coloring while we

used is clearly a lower bound for the minimum number of colors necessary to properly color the whole temporal graph. We color the remaining vertices with a fresh color. This clearly gives a proper sliding Δ-window temporal coloring for the whole temporal graph that uses at most one extra color compared to the optimum. □

5 Conclusion

In this paper we introduced and studied two natural temporal extensions of the classical graph coloring problem, called Temporal Coloring and SW-Temporal Coloring, where Temporal Coloring is the special case of SW-Temporal Coloring, where the sliding window size equals the lifetime of the input temporal graph. For both variants we showed that they are NP-complete even under severe restrictions, in particular even if the number of colors is two, which stands in stark contrast to the static case, where this problem is polynomial-time solvable.

On the positive side, we provided a linear time FPT-algorithm for parameter "number $|V|$ of vertices" and a linear time FPT-approximation algorithm for parameter "vertex cover number of the underlying graph" with an additive error of one. We leave as an open question whether for the latter, we can replace vertex cover number by a structurally smaller parameter.

There are several natural extensions of our problem that one could consider in future work. Considering our motivating example of mobile agents, it would be reasonable to assume that agents do not want to change a channel too frequently. In our model, this would translate to imposing a restriction on the number of color reassignments per vertex, or imposing a minimum time period that each vertex has to wait (after a color change) before it can change colors again. We remark that restricting the number of vertices that may change their color when going from one time slot to the next (which would be a somewhat similar condition as used in "multistage" problems [5, 27, 22]) presumably does not simplify the problem, since in the reduction of the proof of Theorem 4.8 this number is constant (for a constant number of colors k).

References

[1] E. Aaron, D. Krizanc, and E. Meyerson. DMVP: foremost waypoint coverage of time-varying graphs. In *Proceedings of the 40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '14)*, volume 8747 of *Lecture Notes in Computer Science*, pages 29–41. Springer, 2014.

[2] E. C. Akrida, L. Gasieniec, G. B. Mertzios, and P. G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *Journal of Parallel and Distributed Computing*, 87:109–120, 2016.

[3] E. C. Akrida, L. Gasieniec, G. B. Mertzios, and P. G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017.

[4] E. C. Akrida, G. B. Mertzios, P. G. Spirakis, and V. Zamaraev. Temporal vertex covers and sliding time windows. *Journal of Computer and System Sciences*, 107:108–123, 2020.

[5] E. Bampis, B. Escoffier, M. Lampis, and V. T. Paschos. Multistage matchings. In *Proceedings of the 16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT '18)*, volume 101 of *LIPIcs*, pages 7:1–7:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[6] M. Bentert, A.-S. Himmel, H. Molter, M. Morik, R. Niedermeier, and R. Saitenmacher. Listing all maximal k-plexes in temporal graphs. *ACM Journal of Experimental Algorithmics*, 24(1):1–13, 2019.

[7] H. L. Bodlaender, B. M. Jansen, and S. Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.

want to solve the minimization problem here. This can be done by trying out all values for the number k of colors between one and the size of the vertex cover of the underlying graph.

[8] B.-M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

[9] J. F. Buss and J. Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22(3):560–572, 1993.

[10] A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, 2013.

[11] A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada, April 2013.

[12] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[13] A. Casteigts, A.-S. Himmel, H. Molter, and P. Zschoche. The computational complexity of finding temporal paths under waiting time constraints. In *Proceedings of the 31st International Symposium on Algorithms and Computation (ISAAC '20)*, LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. Accepted for publication.

[14] J. Chen, H. Molter, M. Sorge, and O. Suchý. Cluster editing in multi-layer and temporal graphs. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC '18)*, volume 123 of *LIPIcs*, pages 24:1–24:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[15] A. E. F. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics*, 24(4):1694–1712, 2010.

[16] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[17] J. Enright, K. Meeks, G. Mertzios, and V. Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. In *Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science (MFCS '19)*, volume 138 of *LIPIcs*, pages 57:1–57:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019.

[18] T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP '15)*, volume 9134 of *Lecture Notes in Computer Science*, pages 444–455. Springer, 2015.

[19] A. Ferreira. Building a reference combinatorial model for MANETs. *IEEE Network*, 18(5):24–29, 2004.

[20] P. Flocchini, B. Mans, and N. Santoro. Exploration of periodically varying graphs. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC '09)*, volume 5878 of *Lecture Notes in Computer Science*, pages 534–543. Springer, 2009.

[21] T. Fluschnik, H. Molter, R. Niedermeier, M. Renken, and P. Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020.

[22] T. Fluschnik, R. Niedermeier, V. Rohm, and P. Zschoche. Multistage vertex cover. In *Proceedings of the 14th International Symposium on Parameterized and Exact Computation (IPEC '19)*, volume 148 of *LIPIcs*, pages 14:1–14:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019.

[23] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[24] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[25] S. Ghosal and S. C. Ghosh. Channel assignment in mobile networks based on geometric prediction and random coloring. In *Proceedings of the 40th IEEE Conference on Local Computer Networks (LCN '15)*, pages 237–240. IEEE, 2015.

[26] G. Giakkoupis, T. Sauerwald, and A. Stauffer. Randomized rumor spreading in dynamic graphs. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP '14)*, volume 8573 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2014.

[27] A. Gupta, K. Talwar, and U. Wieder. Changing bases: Multistage optimization for matroids and matchings. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP '14)*, volume 8572 of *Lecture Notes in Computer Science*, pages 563–575. Springer, 2014.

[28] A. Himmel, H. Molter, R. Niedermeier, and M. Sorge. Adapting the bron-kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35:1–35:16, 2017.

[29] P. Holme and J. Saramäki, editors. *Temporal Networks*. Springer, 2013.

[30] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[31] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[32] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[33] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.

[34] M. Latapy, T. Viard, and C. Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61:1–61:29, 2018.

[35] J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[36] G. B. Mertzios, O. Michail, and P. G. Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, 2019.

[37] G. B. Mertzios, H. Molter, and V. Zamaraev. Sliding window temporal graph coloring. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '19)*, pages 7667–7674. AAAI Press, 2019.

[38] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science (FOCS '80)*, pages 17–27. IEEE, 1980.

[39] O. Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

[40] O. Michail and P. G. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016.

[41] O. Michail and P. G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72–72, 2018.

[42] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC '78)*, pages 216–226. ACM, 1978.

[43] J. K. Tang, M. Musolesi, C. Mascolo, and V. Latora. Characterising temporal distance and reachability in mobile and online social networks. *Computer Communication Review*, 40(1):118–124, 2010.

[44] C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85 – 89, 1984.

[45] T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

[46] V. G. Vizing. On coloring problems for two-season multigraphs. *Journal of Applied and Industrial Mathematics*, 9(2):292–296, 2015.

[47] F. Yu, A. Bar-Noy, P. Basu, and R. Ramanathan. Algorithms for channel assignment in mobile wireless networks using temporal coloring. In *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems (MSWiM '13)*, pages 49–58. ACM, 2013.

[48] P. Zschoche, T. Fluschnik, H. Molter, and R. Niedermeier. The complexity of finding small separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020.